Machine Learning (2022) 111:3245–3277
https://doi.org/10.1007/s10994-022-06215-7

An adaptive polyak heavy-ball method

Samer Saab Jr.[1] · **Shashi Phoha**[1] · **Minghui Zhu**[1] · **Asok Ray**[1]

Received: 18 October 2021 / Revised: 9 May 2022 / Accepted: 12 June 2022 /
Published online: 18 July 2022

Abstract
The heavy-ball (HB) method has become a well-known practice for large-scale machine learning problems, and it can achieve the fastest local convergence rate when objective functions are smooth and strongly convex using Polyak's optimal hyper-parameters. However, such convergence rates are based on specific uncertain and time-invariant hyper-parameters that limit its potential. In this paper, we propose an adaptive HB that estimates the Polyak's optimal hyper-parameters at each iteration. Our adaptive approach employs the absolute differences of current and previous model parameters and their gradients. Such representation allows for a computationally efficient optimizer. We show that our method guarantees a global linear convergence rate for smooth and strongly convex objective functions. Whereas in the stochastic setting, we show that proposed stochastic algorithm converges almost surely for non-convex smooth functions with bounded gradient. We validate the effectiveness of our method on image classification datasets with no empirical tuning, and its superiority on quadratic and non-convex functions while comparing its performance to the state-of-the-art optimizers.

Keywords Polyak heavy-ball · Gradient descent · Global convergence

Editor: Lam M. Nguyen

✉ Samer Saab Jr.
sys5880@psu.edu

Shashi Phoha
sxp26@psu.edu

Minghui Zhu
muz16@psu.edu

Asok Ray
axr2@psu.edu

[1] The Pennsylvania State University, State College, PA, USA

1 Introduction

First-order stochastic gradient descent (SGD) methods have gained wide popularity in machine learning, as they are well-suited for large-scale problems due to their cheap computational costs while achieving high performance.

Despite many application-specific successes, SGD methods suffer from two limitations. The first limitation, and the most challenging aspect in practical applications of first-order SGD optimizers, is tuning their hyper-parameters, which need to be optimized for maximal performance (Probst et al., 2019). The most common hyper-parameter to tune across all methods is the step-size or the learning rate. Many studies have been implemented on how to select the step-size, which under appropriate constraints can lead to specific convergence properties. For example, constant step-sizes lead to convergence to neighborhoods of local optimum, whereas decreasing step-sizes lead to convergence to the global optimum for convex functions (Ghadimi & Lan, 2013; Gower et al., 2019). However, the tuning of the step-sizes remains to be a tedious task, since their selection ultimately dictates whether the training process will converge or not, and how well the network will eventually perform. Furthermore, the tuning process can be very expensive for very large-scale problems. For example, the GPT-3 network is composed of 175 billion parameters, and was reported to cost an estimated $12 million to train (Floridi & Chiriatti, 2020). The second limitation of SGD methods concerns the rather restrictive conditions to guarantee convergence, and the rates at which they converge. As a result, many first-order SGD methods have been promoted by providing accelerated methods which converge depending on various conditions and assumptions, such as convexity or bounded gradients.

A surge in the developments of optimal convergence rates came about in the past decades to solve large-scale problems (Polyak, 1964; Nesterov, 1983; Lin et al., 2015). Among the most popular methods is the Polyak heavy-ball (HB) method (Polyak, 1964), which can achieve the fastest *local* convergence rate when objective functions are μ-strongly convex and twice continuously differentiable and their gradients are Lipschitz continuous with constant L (Lessard et al., 2016). Polyak showed this through the use of local analysis based on the bounds of the norm of the Hessian of the cost function, which holds globally if the Hessian is constant (Ghadimi, 2015). For a non-convex objective function, it is shown in Zavriev and Kostyuk (1993) that the HB method converges to some stationary point.

The HB method extends the standard GD by adding a momentum term, which incorporates the difference between the current and past parameter iterates, to steer the next parameter iterate towards the solution, as such:

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

where $f(x)$ is the objective function, ∇ is the gradient operator, α_k is the step-size, β_k is the momentum hyper-parameter, and k denotes the iteration number. A local accelerated linear convergence rate of $\frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}$ is guaranteed with the optimal choice of the hyper-parameters; specifically, $\alpha_k^* = \frac{4}{(\sqrt{L}+\sqrt{\mu})^2}$ and $\beta_k^* = \left(\frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}\right)^2$ (Polyak, 1987), which we refer to as Polyak's optimal hyper-parameters throughout this paper. These optimal hyper-parameters require the knowledge of the Lipschitz constant, L, and μ, which are generally inaccessible. Thus, HB would require tuning of its momentum hyper-parameter as well as its step size, which makes it even more burdensome.

Accordingly, adaptive methods (Duchi et al., 2011; Hinton et al., 2012; Kingma and Ba, 2014; Dauphin et al., 2015; Ward et al., 2019) are particularly helpful when training deep neural networks, and are becoming the state-of-the-art (Li and Orabona, 2019). The first popular adaptive method, AdaGrad (Duchi et al., 2011), significantly outperforms the vanilla SGD. From there various extensions have been proposed, such Adam (Kingma and Ba, 2014), which further improved the optimization of deep neural networks (Reddi et al., 2019; Dozat, 2016). The work in Dauphin et al. (2015) shows that adaptive methods can be used to design preconditioners, which help in escaping saddle points. However, several conditions must be met in order to guarantee convergence, such as for RMSProp (Hinton et al., 2012) and Adam, which can be non-convergent even in convex settings (Zou et al., 2019). Furthermore, the variance of such adaptive step-sizes tends to be too large in the early stages of training. Thus, they would require some sort of warmup heuristic, which does not guarantee consistent improvements for various machine learning settings (Liu et al., 2019).

Inspired by the success stories of adaptive methods and the potential of accelerated convergence rate of HB using Polyak optimal hyper-parameters, we propose a novel adaptive HB method[1]. Our proposed method estimates the Polyak's optimal hyper-parameters at each iteration. This task does not require any significant increase in computational complexity.

1.1 Related work

There has been a lot of attention towards accelerated (Beck, 2017; Ghadimi & Lan, 2016) and adaptive optimization techniques (Hu et al., 2009; Huang et al., 2020), which were shown to yield great performance results for large-scale systems, such as deep neural networks (Krizhevsky et al., 2012; Huang et al., 2017). Much of the success of momentum-based optimizers can be attributed to their convergence properties. The convergence analysis of momentum-based optimization methods has been explored under the context of convex (Ghadimi et al., 2015; Ochs et al., 2015) and non-convex (Ochs et al., 2014; Gadat et al., 2018) optimization problems for smooth functions, and non-smooth functions (Mai and Johansson, 2020). Thus in this section, we briefly go over the convergence analyses of selected HB results, and their close variants—our literature review is by no means comprehensive. We then go on to explore the advantages and convergence properties of adaptive optimizers.

1.1.1 Convergence analysis of HB for convex objective functions

The local convergence rate of the HB method was originally established for convex functions near a twice-differentiable local minimum with Lipschitz continuous gradients, and found to have a local accelerated linear rate of convergence when equipped with Polyak optimal hyper-parameters (Polyak, 1964). A global convergence analysis was then presented in Ghadimi et al. (2015), where they showed that if the hyper-parameters of the HB method are chosen within appropriate intervals, and the gradients are Lipschitz continuous,

[1] The main results in this paper are presented in the ICML 2021 workshop on "Beyond First Order Methods in Machine Learning". However, unlike the work in this paper, the objective function is approximated by time-varying positive quadratic function where the proofs are developed accordingly.

then the Cesáro average of the HB method converges at a rate of $\mathcal{O}(1/k)$ when the objective function is convex, and the objective function converges linearly when it is strongly convex. Furthermore, when assuming a strongly convex quadratic objective function, a proof was derived in Lessard et al. (2016), using Lyapunov-like potential functions called integral quadratic constraints, that shows iteration-independent linear rate of convergence when using the Polyak optimal hyper-parameters. More recently, in Scieur and Pedregosa (2020), the Polyak momentum is shown to be asymptotically optimal under the average complexity of all possible inputs to the model, given that the objective function is quadratic with a symmetric positive definite Hessian matrix. This result is shown to be independent of the probability distribution over the inputs.

1.1.2 Convergence analysis of HB for non-convex objective functions

When considering a non-convex objective function, while still assuming Lipschitz continuous gradients, it is shown in Zavriev and Kostyuk (1993) that any trajectory $\{x_k\}$ in the heavy-ball method converges to some stationary point. Similarly, the authors in Ochs et al. (2014) use a monotonically decreasing Lyapunov function for the dynamical system described by the HB method, and show that it is converging.

1.1.3 Convergence analysis of other momentum-based optimizers

Convergence guarantees have also been recently established for other momentum-based optimizers (Hu et al., 2009; Huang et al., 2020; Hendrikx et al., 2020; Cutkosky and Mehta, 2020; Simsekli et al., 2020). For example, the quasi-hyperbolic momentum (QHM) optimizer in Gitman et al. (2019) is shown to converge almost surely for smooth non-convex functions as $\beta_k \to 0$, and ensures local stability near a strict local minimum when the hyper-parameters of QHM are made stationary. Other optimizers have been found to achieve even faster global linear convergence to the optimizer, such as triple momentum (TM) proposed by Van Scoy et al. (2017). However despite the accelerated convergence rate achieved by TM, it requires the burdensome task of tuning four parameters, three of which are momentum hyper-parameters. Another example of an optimizer that achieves faster convergence rate than that of the HB method is the optimizer proposed in Lessard et al. (2016). In fact, the HB and Nesterov accelerated gradient (NAG) serve as special cases of this optimizer, yet its limitation lies in the fact that it requires the tuning of three parameters. However, to the best of the authors' knowledge, there is no published implementation of the TM optimizer nor the optimizer proposed by Lessard *et. al* on neural networks.

1.1.4 Convergence analysis of adaptive gradient methods

The importance of varying the step-size α_k dates back several decades (Robbins and Monro, 1951); where it has been shown that under mild assumptions, global convergence can be achieved when the step-size is square summable, but not summable (Bertsekas, 1997). In Gaivoronski (1994), this rule has been replaced by $\alpha_k \to 0$. Eventually, adaptive methods picked up a lot of attention due to their success in various applications, such as machine learning. Amongst the most popular adaptive optimizers that have seen wide use in everyday machine learning applications are AdaGrad (Duchi et al., 2011), Adadelta, (Zeiler, 2012), RMSProp (Hinton et al., 2012), and Adam (Kingma and Ba, 2014). However, there are several

conditions that one must check to ensure convergence guarantees for adaptive methods, such as for RMSProp and Adam, where Adam can be non-convergent even in convex settings (Zou et al., 2019). It was also shown in Wilson et al. (2017) that to achieve desired performances using adaptive methods, the same amount of hyper-parameter tuning is performed as in non-adaptive methods; challenging the conventional wisdom that adaptive methods require less tuning. However, we duly note our method can perform comparably to the methods compared in Wilson et al. (2017) without hyper-parameter tuning (i.e. $\gamma = 1$) in image classification tasks. The work in Khan et al. (2018) proposes a method that introduces perturbations to the network parameters during gradient evaluations, as well as estimate uncertainty parameters, which can be implemented in Adam. However, this algorithm adds an additional *precision* hyper-parameter that needs to be approximated (e.g. using Bayesian optimization) prior to using the method, as well as require memory to store the uncertainty estimates used in the update rule. Thus, it would require a descent amount of tuning for desired performance. The last-iterate convergence of constrained convex functions for an adaptive heavy-ball method is studied in Tao et al. (2021), where the step-size of the their adaptive HB is updated using an exponential moving average. Using $\beta_{1t} = \frac{t}{t+2}$ and $\beta_{2t} = 1 - \frac{\gamma}{t}$, where t is the epoch number, they could achieve accelarateed convergence. Specifically, their adaptive HB method attains a convergence rate of $\mathcal{O}(\frac{1}{\sqrt{t}})$ as oppose to $\mathcal{O}(\frac{\log t}{\sqrt{t}})$ of SGD. However it remains that two hyper-parameters (α and γ) need to be properly selected for the best performance. On the other hand, attempts at devising stochastic Polyak step-sizes (SPS) have been made for an adaptive SGD optimizer (Loizou & Richtárik, 2020; Berrada et al., 2020; Oberman and Prazeres, 2019; Rolinek and Martius, 2018). The SPS proposed in Loizou and Richtárik (2020) has the form $\gamma_k = \frac{f(x_k)-f^*}{c||\nabla f(x_k)||^2}$, where for machine learning applications they assume $f^* = 0$. Although, as the authors point out, this may seem logical for empirical risk minimization problems, it is unreasonable to assume the knowledge of f^* for majority of optimization problems. Similarly, the work established in Oberman and Prazeres (2019) proposed a SPS of the form $\gamma_k = 2\frac{f(x_k)-f^*}{\mathbb{E}||\nabla f(x_k)||^2}$. This method assumes the knowledge of $\mathbb{E}||\nabla f(x_k)||$, which is unpractical for finite-sum problems with large n (Loizou et al., 2020). Another similar SPS is proposed in Berrada et al. (2020), where $\gamma_k = \min\{\frac{f(x_k)}{||\nabla f(x_k)||^2+\delta}, \eta\}$, however this approach has rather restrictive assumptions on the smoothness of the objective function. The work in Berrada et al. (2020) also provides an SPS method, however without establishing convergence guarantees.

Our proposed adaptive HB method requires the knowledge of only the gradient of the objective function to update its hyper-parameters, and guarantees global convergence. It is also worth mentioning that the work in Ghadimi et al. (2015) tackles the selection of the HB parameters that can yield global convergence. The study provides a sound framework that derives an interval of allowable values for the step-size that depend on the choice of the momentum parameter, while presuming the exact knowledge of L. In addition, for strongly convex functions and linear convergence rate, the values of both L and μ are required. Whereas our algorithm may require tuning of only one parameter that would simultaneously generate α_k and β_k at every iterate interdependently formulated based on the Polyak HB optimal hyper-parameters.

1.2 Contributions

Although adaptive methods are becoming the state-of-the-art and are particularly helpful when training deep neural networks, the restrictive conditions that guarantee convergence of popular adaptive methods, see, e.g., Zou et al. (2019), may limit their application

domains and may be difficult to justify in practice. On the other hand, the accelerated convergence of HB using Polyak optimal parameters may result only in local convergence and/or does not necessarily converge on strongly convex functions (Lessard et al., 2016). It is thus intriguing to ask whether an adaptive optimizer can capitalize on the advantages of adaptive methods and the Polyak optimal convergence rate, while overcoming their limitations. The work in this paper aims to address this question. To that end, we make the following contributions:

- We propose a novel adaptive HB that adopts the formulation of Polyak HB optimal parameters that at worst requires the tuning of a single parameter. To the best of the authors' knowledge, no work has been published on adaptively adopting the Polyak HB optimal hyper-parameters.
- We show that our method guarantees global linear convergence rate for smooth and strongly convex objective functions.
- In the stochastic setting, we show that the proposed stochastic algorithm converges almost surely for non-convex smooth functions provided that the gradient is bounded.
- We also provide an example to show that our adaptive HB can outperform the optimal Polyak HB on a continuously differentiable and strictly convex objective function.

We empirically validate the performance of our method on both convex and non-convex objective functions, including image classification tasks. We use the function presented by Lessard et al. (2016), where HB does not necessarily converge on strongly convex, but not twice differentiable, objective functions when using the Polyak optimal hyper-parameters. We empirically illustrate how our method linearly converges to its optimal solution at a rate faster than the time-invariant Polyak optimal convergence rate. In addition, we empirically illustrate how our method is inherently capable of rejecting gradient noises, leading to robust solutions. We also show how our method outperforms popular adaptive methods on quadratic and the non-convex Beale function in terms of convergence rate. Finally, we demonstrate the competitiveness of our method with popular optimizers on image classification tasks; namely, MNIST, QMINST, CIFAR-10, and CIFAR-100. We find that our approach also enjoys the practical advantage of minimal tuning, and in some cases no tuning at all, as with the image classification tasks.

2 Proposed adaptive HB method

In spirit of proposing adaptive hyper-parameters for HB, we would like to highlight the synergistic coupling bestowed on the adopted Polyak HB optimal hyper-parameters. The core of our approach's success can be accredited to this interdependent coupling, as we propose a method that derives the hyper-parameters simultaneously in an iterative manner according to Polyak HB optimal parameters. First, the quantity $\hat{L}_k$, which emulates the *local* Lipschitz constant L_k of $f(x_k)$, is computed using inner products of current and previous gradients, h_k, and model parameters, Δx_k, as follows:

$$\hat{L}_k \triangleq \gamma \frac{\|h_k\|}{\|\Delta x_k\|} \tag{2}$$

where $h_k \triangleq \nabla f(x_k) - \nabla f(x_{k-1})$, $\Delta x_k \triangleq x_k - x_{k-1}$ and $\gamma \geq 1$. Subsequently, under the context of μ-strongly convex functions, the approximation of μ_k of $\hat{f}_k(x_k)$, denoted $\hat{\mu}_k$, is computed as follows:

$$\hat{\mu}_k \triangleq \frac{\|h_k - \hat{L}_k \Delta x_k\|}{\|\Delta x_k\|}. \tag{3}$$

Both $\hat{L}_k$ and $\hat{\mu}_k$ are then used to approximate the optimal Polyak optimal HB parameters:

$$\alpha_k \triangleq \frac{4}{(\sqrt{\hat{L}_k} + \sqrt{\hat{\mu}_k})^2}, \quad \beta_k \triangleq \left(\frac{\sqrt{\hat{L}_k} - \sqrt{\hat{\mu}_k}}{\sqrt{\hat{L}_k} + \sqrt{\hat{\mu}_k}}\right)^2. \tag{4}$$

If $\hat{L}_k = L$ and $\hat{\mu}_k = \mu$, then α_k and β_k become the optimal Polyak HB parameters.

The proposed AHB algorithm is summarized in Algorithm 1.

Algorithm 1 Adaptive Heavy-Ball

Choose $\gamma \geq 1$
Initialize: $x^- \leftarrow x_{-1}$, $x \leftarrow x_0$ with $x_{-1} \neq x_0$, and get $g^- \leftarrow \nabla f(x_{-1})$
for iteration $k = 1, 2, \ldots$ **do**
  $g \leftarrow \nabla f(x)$
  $\Delta g \leftarrow g - g^-$
  $\Delta x \leftarrow x - x^-$
  $g^- \leftarrow g$
  $x^- \leftarrow x$
  $\hat{L} \leftarrow \gamma \frac{\|\Delta g\|}{\|\Delta x\|}$
  $\hat{\mu} \leftarrow \frac{\|\Delta g - \hat{L}\Delta x\|}{\|\Delta x\|}$
  $\alpha \leftarrow \frac{4}{(\sqrt{\hat{L}} + \sqrt{\hat{\mu}})^2}$
  $\beta \leftarrow \left(\frac{\sqrt{\hat{L}} - \sqrt{\hat{\mu}}}{\sqrt{\hat{L}} + \sqrt{\hat{\mu}}}\right)^2$
  $x \leftarrow x - \alpha g + \beta \Delta x$
end for

Our proposed formulation associated with $\hat{L}_k$ (2) and $\hat{\mu}_k$ (3) is inspired by the Power Iteration Algorithm (PIA) (Mises & Pollaczek-Geiringer, 1929). L and μ are considered as the largest and smallest eigenvalues of the Hessian matrix, $\nabla^2 f(x)$, where $\nabla f(x) \approx \nabla^2 f(x)x + b$. Thus, $h_k \approx \nabla^2 f(x_k)\Delta x_k$ and $\frac{h_k^T h_k}{\Delta x_x^T \Delta x_k} \approx \frac{\Delta x_x^T \left(\nabla^2 f(x_k)\right)^2 \Delta x_k}{\Delta x_x^T \Delta x_k}$. The latter can be thought of as applying two iterations of PIA for each k, which could suffice as a good approximation in the case where the largest eigenvalue is significantly larger than the rest. For strongly convex setting where $\nabla^2 f(x)$ is presumed to be a positive-definite matrix, then by similarly applying PIA to $h_k - \hat{L}_k I$, where the eigenvalues of $\nabla^2 f(x) - LI$ become all negative and the smallest eigenvalue of $\nabla^2 f(x)$ would be the eigenvalue of $\nabla^2 f(x) - LI$ with the largest magnitude.

Remark 1 Perfect estimation of L and μ may not be desired under this context due to the limitation of Polyak optimal HB in achieving global convergence; e.g., rippling behavior near the optimum.

3 Global convergence for convex objective functions

In this section we present the convergence analysis of our proposed adaptive HB for convex objective functions.

We first formally present the definitions that are used for attaining convergence.

Definition 1 Function $f(x)$ is L-smooth if it is continuously differentiable, and its gradient is Lipschitz continuous with constant L, i.e., for all $x, y \in \mathbb{R}^d$:

$$||\nabla f(y) - \nabla f(x)|| \leq L||y - x||.$$

Definition 2 A continuously differentiable function $f(x)$ is called strongly convex on $\mathbb{R}^d$, if there exists a constant $\mu > 0$ such that for any $x, y \in \mathbb{R}^d$ we have (Nesterov, 2003)

$$f(y) - f(x) \geq \langle y - x, \nabla f(x)\rangle + \frac{\mu}{2}||y - x||^2. \tag{5}$$

We first present some useful properties in Proposition 1 that relate γ to $\hat{L}_k$, $\hat{\mu}_k$, α_k and β_k.

Proposition 1 *Suppose that $f(x)$ is μ-strongly convex and L-smooth. Then, $L \geq \mu$, and the algorithm (1) with hyper-parameters γ, α_k and β_k in (4), guarantees the following properties for all $k \geq 0$:*

(P1) $\gamma\mu \leq \hat{L}_k \leq \gamma L$.

For all $\gamma \geq \frac{L}{\mu} + \sqrt{\frac{L^2}{\mu^2} - 1}$, *we have*

(P2) $\sqrt{\gamma^2\mu^2 + \mu^2 - 2\gamma\mu L} \leq \hat{\mu}_k \leq (\gamma + 1)L$.

(P3) $\beta_k \in (0, 1)$, *and*

$$\frac{4}{(\sqrt{\gamma} + \sqrt{\gamma + 1})^2 L} \leq \alpha_k \leq \frac{4}{(\sqrt{\gamma\mu} + (\gamma^2\mu^2 + \mu^2 - 2\gamma\mu L)^{\frac{1}{4}})^2}.$$

(P4) $\hat{\mu}_k$ *is an increasing function of* γ, *and* α_k *is a decreasing function of* γ.

(P5) $\lim_{\gamma\to\infty} \hat{\mu}_k = \lim_{\gamma\to\infty} \hat{L}_k = \infty$, *and*

$\lim_{\gamma\to\infty} \alpha_k = \lim_{\gamma\to\infty} \beta_k = 0$.

Theorem 1 *Suppose that $f(x)$ is μ-strongly convex and L-smooth. Then, there exists a $\gamma_0 \geq \max\left\{4\left(\frac{L}{\mu}+\sqrt{\frac{L^2}{\mu^2}-1}\right), \frac{4(L+1)}{\mu}\right\}$ such for $\gamma > \gamma_0$, the sequence $\{x_k\}$ generated by algorithm (1) with step-size α_k and hyper-parameter β_k in (4), satisfies*

$$f(x_k) - f(x^*) \leq q^k(f(x_0) - f(x^*)) \tag{6}$$

where $q \in (0, 1)$, and x^* is the unique optimal solution that minimizes $f(x)$.

The proofs of Proposition 1 and Theorem 1 are included in the appendix.

Theorem 1 states that the algorithm globally converges linearly to the unique optimal point x^* for strongly-convex functions and for $\gamma > \gamma_0$. However, in our empirical results, we use $\gamma = 1$ in all the image classification datasets under consideration to show that our proposed method may not require tuning yet leading to competitive performance with the state-of-art optimizers.

It is shown Ghadimi et al. (2015) if $\beta \in [0, 1)$ and $\alpha \in \left(0, \frac{2(1-\beta)}{L}\right)$, then global convergence is attained for convex and smooth functions where the Cesáro average of the iterates converges to the optimum at a rate of $\mathcal{O}(1/k)$. Say that $L = 5$ and $\mu = 1$. Then, Polyak optimal hyper-parameters become $0 < \alpha^* = 0.382 < 2/L$ and $0 < \beta^* = 0.146 < 1$. However, $\alpha^* > \frac{2(1-\beta^*)}{L} = 0.34$. Therefore, a global convergence cannot be guaranteed for convex and smooth whenever optimal Polyak hyperparameters are used. On the other hand, for sufficiently large γ, with a lower bound given in Theorem 1, both α_k and β_k decreases, see, e.g., (P5) of Proposition 1. The latter indicates that the conditions of the global convergence (Ghadimi et al., 2015) can be satisfied for convex and smooth functions.

In addition, Ghadimi et al. (2015) provides the following condition for linear global convergence: for $\alpha \in \left(0, \frac{2}{L}\right)$, the momentum parameter should satisfy $0 \leq \beta < \frac{1}{2}\left(\frac{\mu\alpha}{2}+\sqrt{\frac{\mu^2\alpha^2}{4}+4(1-\frac{\alpha L}{2})}\right)$. Our results in Theorem 1 presents similar convergence results for the proposed adaptive HB provided that $\gamma > \gamma_0$. However, for both settings of convex and strongly convex functions, the knowledge of L and μ is required to find the bounds provided in Ghadimi et al. (2015), which is not practical. Consequently, in practice, tuning of α and β would be required whereas our adaptive optimizer would require tuning of only one parameter, γ. Furthermore, the example of a convex smooth function provided in Section 4.3 shows that our proposed adaptive HB outperforms both the time-invariant HB and time-varying HB satisfying the conditions presented in Ghadimi et al. (2015).

On another note, in exceptional cases where x is a scalar, then with $\gamma = 1$, we will have $\hat{\mu}_k = 0$ and thus β_k becomes zero. Consequently, the proposed method would become adaptive gradient descent with significantly large $\alpha_k = \frac{4}{\hat{L}_l}$, which may deteriorate the global convergence. Therefore, for scalar x, it is recommended to have $\gamma > 1$.

4 Stochastic adaptive HB

The stochastic optimization problem is of the form

$$\min_{x\in\mathbb{R}^d} F(x) \triangleq \mathbb{E}_{\zeta}[f(x, \zeta)]. \tag{7}$$

We denote by $\mathscr{G}(x_k;\zeta_k)$ a stochastic gradient of $F(x)$ at x_k depending on a random variable ζ_k representing data sampled from some unknown probability distribution such that $\mathbb{E}[\mathscr{G}(x_k;\zeta_k)] \in \partial F(x_k)$, where $\partial F(x)$ denotes the set of gradients of F at the point x.

Our proposed stochastic adaptive heavy-ball (SAHB) method is given by

$$x_{k+1} = x_k - \hat{\alpha}_k \mathscr{G}(x_k;\zeta_k) + \hat{\beta}_k(x_k - x_{k-1}) \tag{8}$$

where $\hat{\alpha}_k = \min\left\{\alpha_k, \frac{C}{(k+1)^\nu}\right\}$, $\hat{\beta}_k = \min\left\{\beta_k, \frac{1}{(k+2)^\nu}\right\}$, $0.5 < \nu \leq 1$, $C > 0$, and α_k and β_k are defined in (4). In addition, we make the following modification:

$$\hat{L}_k \triangleq \gamma \frac{\|\hat{h}_k\|}{\|\Delta x_k\|} \tag{9}$$

where $\hat{h}_k \triangleq \mathscr{G}(x_k;\zeta_k) - \mathscr{G}(x_{k-1};\zeta_{k-1})$, $\Delta x_k \triangleq x_k - x_{k-1}$ and $\gamma \geq 1$. We compute $\hat{\mu}_k$ as follows:

$$\hat{\mu}_k \triangleq \frac{\|\hat{h}_k - \hat{L}_k \Delta x_k\|}{\|\Delta x_k\|}. \tag{10}$$

Both $\hat{L}_k$ and $\hat{\mu}_k$ are then used to approximate the optimal Polyak optimal HB parameters in (4).

The following conditions hold for F defined in (7) and the stochastic gradient:

Assumption 1 F is differentiable and ∇F is Lipschitz continuous, i.e., there exists a constant L such that

$$\|\nabla F(x) - \nabla F(y)\| \leq L\|x - y\|, x, y \in \mathbb{R}^d.$$

Assumption 2 F is bounded below and $\|\nabla F(x)\|$ is bounded above; i.e., there exist F_* and G such that

$$F(x) \geq F_*, \|\nabla F(x)\| \leq G, x \in \mathbb{R}^d.$$

Assumption 3 For $k = 0, 1, 2, ...,$ the stochastic gradient $g_k \triangleq \mathscr{G}(x_k;\zeta_k) = \nabla F(x_k) + \zeta_k$, where the random noise ζ_k satisfies

$$\mathbb{E}_k[\zeta_k] = 0, \mathbb{E}_k[\|\zeta_k\|^2] \leq D a.s.$$

where $a.s.$ refers to almost surely, $\mathbb{E}_k[.]$ denotes expectation conditioned on $\{x_0, g_0, ..., x_{k-1}, g_{k-1}, x_k\}$, and D is a constant.

Assumption 4 The stochastic gradient is Lipschitz continuous, i.e., there exists a constant L_G such that

$$\|\hat{h}_k\| \leq L_G\|x - y\|, x, y \in \mathbb{R}^d.$$

Assumptions 1–3 are the ones used in Gitman et al. (2019). However, we additionally assume that the stochastic gradient is Lipschitz continuous. The following theorem, Theorem 2, can be considered as a corollary to Theorem 1 in Gitman et al. (2019) with $v_k = 1$, which can be applied to non-convex functions.

Theorem 2 *Under Assumptions* 1–4, *the sequence* $\{x_k\}$ *generated by the SAHB algorithm described by* (9)-(10) *and* (8), *satisfies*

$$\liminf_{k\to\infty} \|\nabla F(x_k)\| = 0 a.s. \tag{11}$$

In addition, we have

$$\limsup_{k\to\infty} F(x_k) = \limsup_{k\to\infty, \|\nabla F(x_k)\|\to 0} F(x_k) a.s. \tag{12}$$

The proof of Theorem 2 *is included in the appendix.*

5 Results and discussion

In this section we aim to validate the proposed adaptive HB against other known optimizers by studying its convergence rate, overall performance, and robustness. To demonstrate the reduced tuning efforts that our method offers, we only tune the proposed optimizer where it is reasonable to do so. To that end, we tune our method on one-dimensional and two-dimensional functions, where tuning is cheap. We also show the effect of tuning our method on an example including quadratic cost functions. In large-scale systems however, tuning is computationally expensive, and thus we implement our optimizer on a deep neural network with no tuning (by setting $\gamma = 1$ of Eq. (2)) involved.

First, we show the superiority of the convergence rate of our proposed optimizer over the optimal HB on a positive-definite quadratic function, and the strongly convex function given in Lessard et al. (2016), which was shown to trap the optimal HB in a limit cycle in Sect. 5.2. Then we compare our adaptive HB with a time-invariant and time-varying HB on positive semi-definite quadratic functions We further evaluate the robustness and convergence rate of our adaptive HB optimizer against other known optimizers on the non-convex Beale function in Sect. 5.4. Lastly, we evaluate the optimizer's performance against popular optimizers on MNIST, QMNIST, CIFAR-10, and CIFAR-100 (Krizhevsky et al., 2009) in Sect. 5.5.

It is important to note that in our toy examples (Lessard function, quadratic and Beale functions), we consistently run all optimizers under consideration through all the samples in our training set to do a single update for the weight, x_k, in every iteration, k. On the other hand, in our image classification examples, we also consistently run all optimizers under consideration while using only one minibatch or one subset of the training set to do the update for the weight in every iteration. In addition, to be consistent with the other optimizers, we use AHB throughout all our experiments except for in the case where we compare our AHB and our Stochastic AHB (SAHB) on Beale function with noisy gradients.

Our codes for all experiments will be made publicly available.

5.1 Positive-definite quadratic function

Although the fastest local convergence rate is attained when objective functions are continuously differentiable and strictly convex using Polyak's optimal hyper-parameters, this example is designed to show that this fact is constrained to fixed hyper-parameters. In particular, we demonstrate that our proposed method with adaptive hyper-parameters can outperform

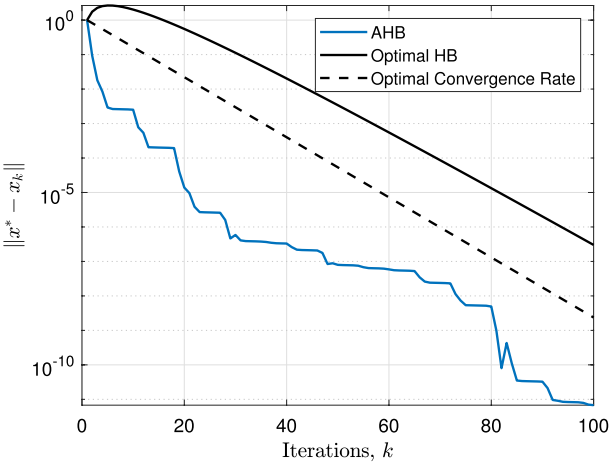

Fig. 1 Visualization of the convergence rate of our optimizer versus optimal HB via the norm of the error at every iteration, k. The smoothness of the optimal HB curve is mostly due to the fact that the step size and the momentum parameter are constants unlike our proposed AHB

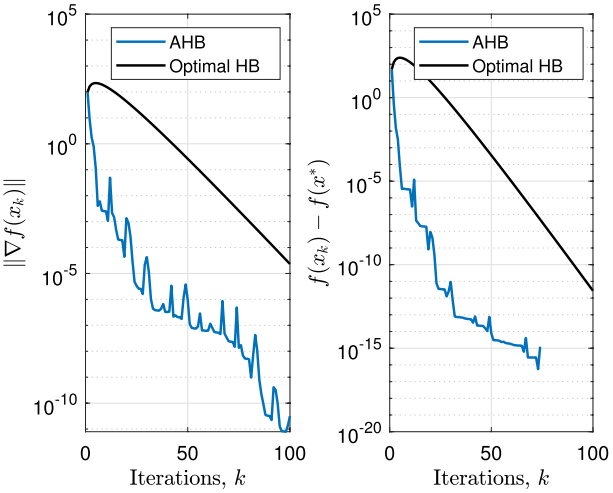

Fig. 2 Visualization of the convergence rate of our optimizer versus optimal HB via the norm of the gradient and $f(x_k) - f(x^*)$ at every iteration, k. For $k \geq 74$, $f(x_k) - f(x^*) < 10^{-20}$ for the proposed AHB method

the optimal Polyak heavy-ball method for a continuously differentiable and strictly convex function.

We consider the following objective function

$$f(x) = \frac{1}{2}x^T Ax - b^T x \tag{13}$$

where

$$A = \begin{bmatrix} 99 & 1 \\ 1 & 1 \end{bmatrix}, b = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, x_{-1} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \text{and } x_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

We use the largest and smallest eigenvalues of A as L and μ, respectively, to compute Polyak HB optimal parameters with corresponding convergence rate of

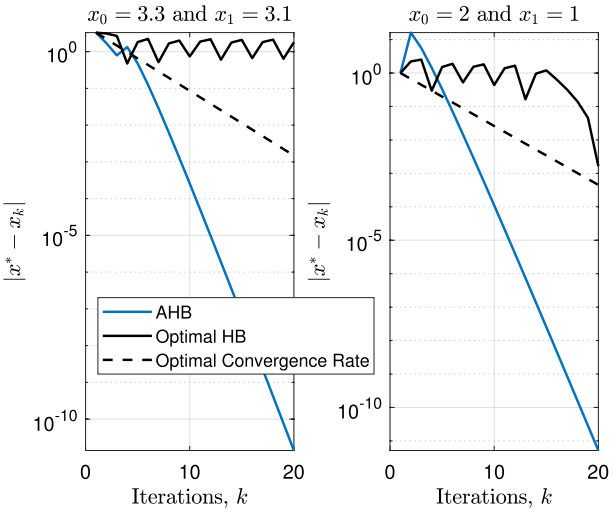

Fig. 3 Visualization of the convergence rate of our optimizer versus optimal HB via the norm of the error at every iteration, k. Except for the first few iterations, the curve of AHB is smooth due to the large rate of convergence at every iteration

$\theta \triangleq \frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}} = 0.818$. For our proposed HB, we use $\gamma = 1.8$ in (2). Figure 1 shows $\|x^* - x_k\|$ for both Optimal HB and the proposed AHB, and also $\|x^* - x_0\|\theta^k$. Figure 2 shows $\|\nabla f(x_k)\|$ (left) and $f(x_k) - f(x^*)$ for both methods. This simple example demonstrates that our adaptive optimizer can outperform the optimal Polyak HB for continuously differentiable and strongly convex function. Of note, the optimal HB requires the exact knowledge of L and μ, whereas our proposed AHB for this specific scenario requires the tuning of γ in order to outperform the optimal Polyak HB.

5.2 Lessard's problem

In this section, we tackle the one-dimensional strongly convex function, $f(x)$, presented in Lessard et al. (2016), which can lead to a non-convergent solution using Polyak optimal hyper-parameters. The function's gradient, $\nabla f(x)$, is continuous and monotone but not continuously differentiable, and is given by the following equation:

$$\nabla f(x) = \begin{cases} 25x & \text{if } x < 1 \\ x + 24 & \text{if } 1 \leq x < 2 \\ 25x - 24 & \text{if } x \geq 2 \end{cases} \tag{14}$$

It is important to note that the optimal local convergence rate, $\frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}$, using Polyak optimal hyper-parameters requires that $\nabla f(x)$ is continuously differentiable, which is not the case of the function in (14).

Although $\nabla f(x)$ is not continuously differentiable, one can readily observe that $L = 25$ and $\mu = 1$. Lessard *et. al* showed that if the initial conditions are chosen to be within the interval $3.07 \leq x_0 \leq 3.46$, then HB with its optimal parameters gets stuck in a limit cycle with oscillations that never damp out. Figure 3 (upper plot) shows how the error $|x_k - x^*|$ using Polyak optimal HB parameters oscillates when the initial conditions are set to $x_0 = 3.3$ and $x_1 = 3.1$, whereas the error corresponding to our adaptive HB continues to

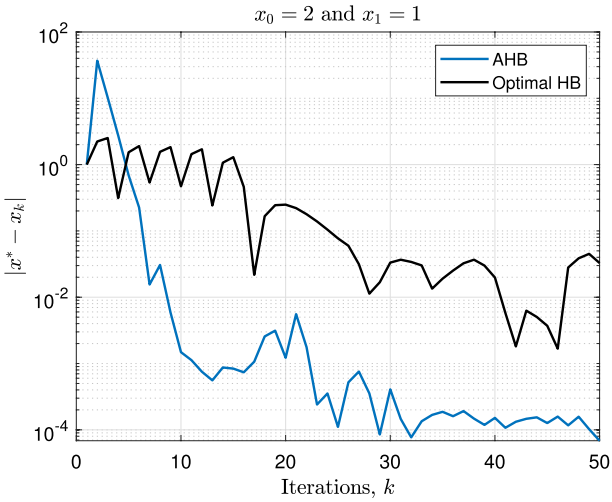

Fig. 4 The norm of the error produced by the proposed adaptive HB versus the optimal HB in the presence of noisy gradients

decrease with no oscillations. We also choose initial conditions that lay outside the aforementioned interval to compare our method with the optimal HB, specifically we set the initial conditions to $x_0 = 2$ and $x_1 = 1$. In all cases, our method with its *time-varying* hyper-parameters and with $\gamma = 2$ demonstrates a superior convergence rate, which is evident after the fifth iteration, even when compared to the optimal (local) convergence rate of $\frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}$ achieved by the Polyak optimal *time-invariant* hyper-parameters.

The optimal convergence rate shown in Fig. 3 is simply the plot of $\theta^k||x_0 - x^*||$, where $\theta = \left(\frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}\right) = \frac{2}{3}$ for this problem. When fitting $c\theta^k||x_0 - x^*||$ to the error plots achieved by our method, we find $\theta = 0.19$. This result restates that our method can achieve a faster convergence rate (e.g.,this problem) than the Polyak optimal HB. It is important to note that optimality of Polyak HB requires the gradient to be continuously differentiable, which does not apply to (14).

Additionally, we test our method and the optimal HB method on the Lessard problem with noisy gradients. The objective of this experiment is to demonstrate how our method is inherently capable of rejecting measurement noise, leading to robust performance. We add zero-mean white noise, ξ_k, to the gradient sampled from a normal distribution with a standard deviation of 0.1. The initial conditions are set to $x_0 = 2$ and $x_1 = 1$, and we set γ of Eq. (2) to 1.3. It is clear from Fig. 4 that our proposed adaptive HB outperforms the optimal HB, as the error continues to decrease within the first 31 iterations to reach an error of about 1×10^{-4}. The norm of the error produced by the optimal HB get stuck fluctuating between 10^{-1} and 10^{-3} from the 25th iteration onward. This suggests the the optimal HB was incapable in further rejecting or suppressing the measurement noise, whereas the continuously deceasing error generated using our adaptive HB suggests that it is capable of partially suppressing the measurement noise. After a finite number of iterations, α_k and $x_k - x_{k-1}$ converge to a steady-state solution.

The way in which the noise gets suppressed is explained in the following reasoning. The additional noise drives $\hat{L}_k = \gamma\sqrt{\frac{\Delta {g_k}^T \Delta g_k}{\Delta x_k^T \Delta x_k}}$, where $g_k = \nabla f(x_k) + \xi_k$, to larger values, which causes α_k to decrease towards zero—a needed trend when dealing with noise.

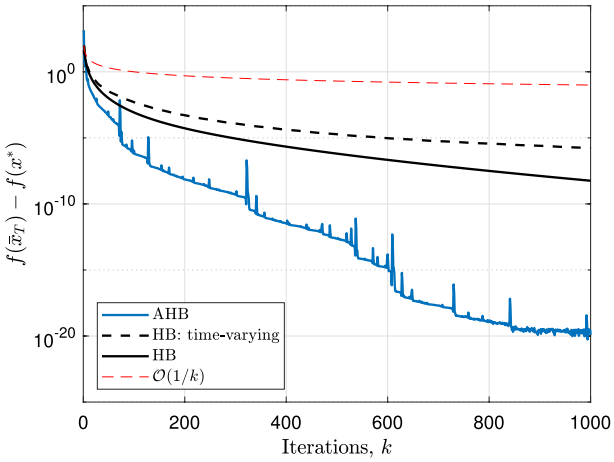

Fig. 5 Comparison of the progress of the objective values with $A \in \mathbb{R}^{50\times 50}$ evaluated at the Cesáro average of the iterates of the three heavy-ball methods under study, namely: the proposed adaptive heavy-ball with $\gamma = 1.2$, the heavy-ball with time-varying hyper-parameters $\alpha_k = \frac{1}{L(k+1)}$ and $\beta_k = \frac{k}{k+2}$, and the one HB with $\alpha = \frac{1}{L}$ and $\beta = 0.5$

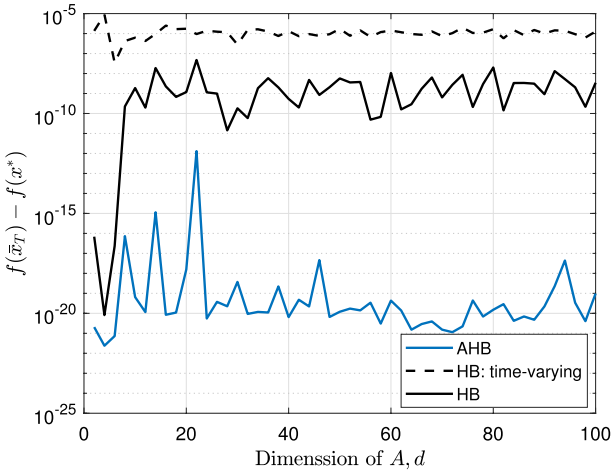

Fig. 6 Comparison of the objective values with $A \in \mathbb{R}^{d\times d}, 2 \le d \le 100$, evaluated at the Cesáro average at the iterate $k = 20d$ of the three heavy-ball methods under study

5.3 Positive semi-definite quadratic functions

This example illustrates the performance of our proposed HB against a time-invariant and time-varying HB satisfying the conditions of the work in Ghadimi et al. (2015). We consider a quadratic objective function with positive semi-definite quadratic matrix, $A \in \mathbb{R}^{d\times d}$. We generate random matrices for $d = 2, 4, \ldots, 98, 100$ with rank equals to $\frac{d}{2}$ as follows: $A = \sum_{j=1}^{\frac{d}{2}} \xi_j \xi_j^T$, where each element of $\xi_j \in \mathbb{R}^d$ is sampled from a uniform distribution over the interval [0, 1]. Similarly, each element of the initial guess $x_0 \in \mathbb{R}^d$ is sampled from a normal distribution, $\mathcal{N}(0, 1)$. For each d we run $20d$ iterations.

As reflected in the example given in Ghadimi et al. (2015), for the time-invariant HB, we choose $\alpha = \frac{1}{L}$ and $\beta = 0.5$ and for time-varying HB, we choose $\alpha_k = \frac{1}{L(k+1)}$ and $\beta_k = \frac{k}{k+2}$. However, it is important to note that the selected step-size requires the knowledge of the largest eigenvalue of A, L, for each dimension d. On the other hand, for the

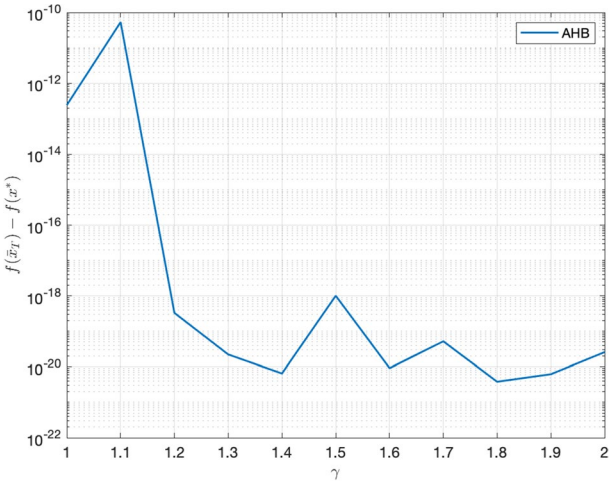

Fig. 7 Comparison of the objective values with $A \in \mathbb{R}^{50\times 50}$ evaluated at the Cesáro average at the iterate $k = 1000$ of the proposed adaptive heavy-ball method for different values of γ

proposed method, we assume no knowledge about A or L, and fix $\gamma = 1.2$ for all dimensions of A.

Figure 5 illustrates the progress of the objective values evaluated at the Cesáro average of the iterates of the three heavy-ball methods under consideration for $d = 50$, that is, $A \in \mathbb{R}^{50\times 50}$. As a reference, we also include $\mathcal{O}(1/k)$ upper-bound. Whereas in Fig. 6, we show the values of $f(\bar{x}_T) - f(x^*)$ at the very last iteration, $k = 20d$, for the three methods, where $\bar{x}_T = \frac{1}{T+1}\sum_{k=0}^{T} x_k$ is the Cesáro average of the iterates. Figure 7 shows that our method converges for $\gamma \geq \gamma_0$ and for this example $\gamma_0 = 1$. It presents the objective values for different values of γ for $d = 50$, that is, $A \in \mathbb{R}^{50\times 50}$, at the very last iteration, $k = 1,000$.

By examining Figs. 5 and 6, the proposed adaptive HB with a fixed $\gamma = 1.2$ for all dimensions outperforms the other two methods that assume the knowledge of L associated with each dimension, d. Also, by examining Fig. 7, we find that even when increasing the value of γ, the superiority of the proposed adaptive HB is preserved.

We emphasize that selection of the hyper-parameters for the HB method without the knowledge of the largest eigenvalue, L, can be indeed tricky. For the quadratic functions generated in this example, the largest L is about 1.3×10^3 at $d = 100$, and the smallest L is about 3.2×10^{-1} at $d = 3$ with a standard deviation of about 380. If L is under estimated, then the step-size becomes too large and that can lead to divergence. Whereas if L is over estimated, then this would lead to smaller values of the step size yielding slower convergence. Consequently, unlike our method, tuning of a non-adaptive HB can become burdensome.

5.4 Non-convex beale function

In this section we aim to compare the convergence and convergence rate of our optimizer when compared to other popular optimizers when solving the non-convex Beale function, which is listed as one of the 175 benchmark test functions for optimization algorithms (Jamil & Yang, 2013). The Beale function is a 2-dimensional non-convex

Table 1 Convergence characteristics of the proposed optimizer in comparison to known optimizers when tested on the Beale function

Optimizer	Wins	Times converged	Average steps
AHB	**801**	**915**	**175.52**
SGD	0	27	992.94
SGDm	34	319	719.11
NAG	12	569	575.23
RMSProp	3	86	933.54
Adagrad	9	346	772.11
Adam	69	535	586.37

Bold reflect the best performing metric values

function with one global minimum $f(x^*) = 0$ at $x^* = (3, 0.5)$. The function is given by: $f(x, y) = (1.5 - x + xy)^2 + (2.25 - x + xy^2)^2 + (2.625 - x + xy^3)^2$.

We conduct 1,000 independent runs where we randomly sample each element of the initial value, x_0, from a uniform distribution over the interval [0, 4). The convergence rate is evaluated by recording the number of steps, k, for any optimizer it took to reach the solution within a difference in error of $f(x_k) - f(x^*) \leq 10^{-5}$. A *win* is awarded to an optimizer if it converges to a solution *first* within the first 1, 000 iterations, if no optimizer converges then none of the optimizers are rewarded.

We compare the proposed adaptive HB with stochastic gradient descent (SGD), SGD with momentum (SGDm) (Qian, 1999), Nesterov's accelerated gradient (NAG) method (Nesterov, 1983), RMSProp (Hinton et al., 2012), AdaGrad (Duchi et al., 2011), Adam (Kingma and Ba, 2014) and AdamW (Loshchilov and Hutter, 2017).

We provide a brief description of the adaptive methods under consideration. Adaptive gradient algorithm (AdaGrad) is a modified stochastic gradient descent algorithm with per-parameter learning rate. That is, the learning rate is a diagonal matrix with elements equal to $\frac{\eta}{\sqrt{G_{i,i}}}$, where $G = \sum_{j=1}^{k} g_j g_j^T$, g_j is the gradient at iteration k, and η is a step size. The idea is based on increasing the learning rate for sparser parameters and decreasing the learning rate for ones that are less sparse. Root Mean Square Propagation (RMSProp) divides the learning rate by a running average of the magnitudes of recent gradients. Adaptive Moment Estimation (Adam), which is considered as an update to RMSProp, uses running averages of both the gradients and the second moments of the gradients. Adam with decoupled weight decay (AdamW) is basically a modification of Adam where the weight decay is decoupled from the optimization steps taken with respect to the loss function.

The learning rates of all these optimizers were chosen after sweeping over the values $\{1, 0.5, 0.1, 0.05, 0.01, 0.005, 0.001\}$. The learning rates that are selected for SGD, SGDm, NAG, RMSProp, Adagrad, and Adam are 0.01, 0.01, 0.001, 0.01, 0.5, and 0.5, respectively. The momentum factor used for SGDm is the standard value of 0.9, and the standard values of $\beta_1 = 0.9$ and $\beta_2 = 0.99$ are used for Adam. As for our adaptive HB, we only conducted a hyper-parameter search for γ of Eq. 2, where we sweep over the values: $\{1, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2\}$. The value taken for all hyper-parameter searches was the one that returned the largest number of times each optimizer converged. The value for γ of Eq. 2 for our adaptive HB was chosen to be 1.2.

Clearly, from Table 1, the proposed adaptive HB outperforms the rest by converging to the global minimum the largest number of times, specifically the adaptive HB converges 801 times with the fastest rate averaging at 175.52 iterations to converge. Not to mention,

Table 2 Average of $\min_{0\leq k\leq 1000} \|\nabla f(x_k)\|^2$ over 10^3 independent runs when tested on the Beale function with noisy gradient

Method	$\sigma = 0$	$\sigma = 10^{-2}$	$\sigma = 10^{-1}$	$\sigma = 1$
AHB	$\mathbf{3.4 \times 10^{-29}}$	1.1×10^{-4}	9.0×10^{-3}	0.26
SAHB	1.2×10^{-14}	$\mathbf{8.8 \times 10^{-5}}$	$\mathbf{3.8 \times 10^{-3}}$	**0.02**

Bold reflect the best performing metric values

our adaptive HB converges 91.5% of all random runs, which is indicative of the method's robustness.

Remark 2 Although we mostly study the performance of popular adaptive optimizers, it is important to note it is still possible to obtain monotonic decreasing function values that in the non-convex setting, even with a fixed hyper-parameter. This can be accomplished by following the negative curvature direction (see, e.g., Carmon et al., 2017; Yu et al., 2018).

5.4.1 AHB versus SAHB in presence of noisy gradient

In this experiment we add zero-mean white Gaussian noise, $\mathcal{N}(0, \sigma^2)$, to the gradient of the Beale function and we run 10^3 independent runs using different values of $\sigma \in \{0, 10^{-2}, 10^{-1}, 1\}$. We use $\gamma = 1.2$, $x_{-1} = [0\ \ 0]^T$, and $x_0 = [1\ \ 1]^T$.

We consider $\min_{0\leq k\leq 1000} \|\nabla f(x_k)\|^2$ and compute its average over the 10^3 independent runs. For AHB, we use α_k and β_k as defined in (4) where as for SAHB, we use $\hat{\alpha}_k = \min\left\{\alpha_k, \frac{C}{(k+1)^\mu}\right\}$, $\hat{\beta}_k = \min\left\{\beta_k, \frac{1}{(k+2)^\mu}\right\}$, as defined in (8) with $C = 1$ and $\mu = 0.5001$.

Examining Table 2, we conclude without the addition of noise, AHB outperforms SAHB, otherwise SAHB outperforms AHB.

Remark 3 if $\alpha_k = \frac{C}{(k+1)^\mu}$ and $\beta_k = \frac{1}{(k+2)^\mu}$, then an HB using such hyper-parameters would require small values of C to converge; e,g, $C \leq 0.1$ and for such setting, this algorithm converges much slower than SAHB or AHB.

5.5 Image classification

We train several existing networks with random initializations on the MNIST, QMNIST, CIFAR-10, and CIFAR-100 image classification datasets. We use these image classification tasks to demonstrate that our optimizer is well-suited for deep neural networks when compared with other popular optimizers. For these tasks, essentially no tuning was performed on our method (i.e. $\gamma = 1$ of Eq. 2). To avoid any divisions by zero, we add a fixed ϵ to the denominator of equations (2) and (3) in all of our image classification tasks to avoid singularities. It is important to note that excessive tuning was conducted on the other three optimizers under consideration. We point out that we compute the gradient at every batch (which is equivalent to an optimization step), rather than a full gradient descent over the entire dataset. Thus, every epoch takes the average accuracy or loss over the total number of batches within each epoch, as commonly done with the other optimizers that we compared against. In addition, we used a common batch sample used by the optimizers under consideration, although recent studies have pointed out that the performance of DNNs is heavily dependent on how well the mini-batch samples are selected (Song et al., 2020; Bakirov & Gabrys, 2021).

Table 3 The average MNIST (top) and QMNIST (bottom) test accuracy results obtained over the last 10 epoch for all optimizers

AHB	SGDm	NAG	Adam	AdamW
98.79%	99.03%	99.13%	**99.17%**	97.77%
±0.03%	±0.02%	±0.01%	**± 0.02%**	±0.03%
98.71%	98.81%	**98.95%**	98.93%	97.61%
±0.02%	±0.02%	**± 0.01%**	±0.01%	±0.04%

Bold reflect the best performing metric values

5.5.1 MNIST/QMNIST

The MNIST dataset is a 10-class image classification dataset composed of 60, 000 and 10, 000 training and testing grey-scale images of hand-written digits, respectively. The QMNIST dataset extends the MNIST testing set, resulting in 60, 000 testing images. We compare our proposed optimizer with SGDm, NAG, Adam, and AdamW. The neural network chosen for this problem is the conventional convolutional neural network (CNN) as designed in Koehler (2020), which includes two convolutional layers with kernel size 5, one fully-connected hidden layer, and a proceeding fully-connected layer of 50 neurons connecting to the output. The activation function chosen is the ReLU function. We run our networks for 200 epochs over 5 random (seeds) initializations of the network parameters, and use a batch size of 64.

The learning rates for the optimizers are chosen by conducting a random search over the values $\{1, 0.1, 0.5, 0.01, 0.05, 0.001, 0.005, 0.0001, 0.0005\}$ for all optimizers, and choose the value that returns the highest validation accuracy. For SGDm, we choose a learning rate value of 0.01, with the standard momentum factor of 0.9. For NAG we choose a learning rate of 0.01 and $\beta = 0.9$. For Adam, we choose a learning rate of 0.0005 with the standard values of $\beta_1 = 0.9$ and $\beta_2 = 0.99$. Lastly, for AdamW, we choose a learning rate of 0.0005 with the standard values of $\beta_1 = 0.9$ and $\beta_2 = 0.99$, and a weight decay value of 1. Our method does not require the selection of a learning rate, as it automatically updates its hyper-parameters at every iteration.

Our results are summarized in Table 3, which includes the average test accuracies for all optimizers over the last 10 epochs. The best results are highlighted in bold. Our method's performance is comparable to the other optimizers in these image classification tasks. We emphasize the fact that no tuning was performed on our method, and one would expect potentially outperforming the other optimizers if the hyper-parameter γ of Eq. (2) were tuned.

5.5.2 CIFAR-10/100

The CIFAR-10 and CIFAR-100 datasets are composed of 50, 000 and 10, 000 natural training and testing images, all with dimensions of 32×32, each with 10 and 100 classes, respectively.

We use the tuning hyper-parameters set in Zhang et al. (2019) for SGDm and AdamW, and run the proposed CIFAR experiments using a Resnet-18 (He et al., 2016) for 200 epochs over three different seeds using a batch size of 128. We also compare against Adam, which is tuned in-house, for further comparison. The learning rate for Adam is chosen by conducting a random search over $\{1, 0.1, 0.5, 0.01, 0.05, 0.001, 0.005, 0.0001, 0.0005\}$ and choose the value that returns the highest validation accuracy. We tune all optimizers on CIFAR-10, then use the same hyper-parameters on CIFAR-100.

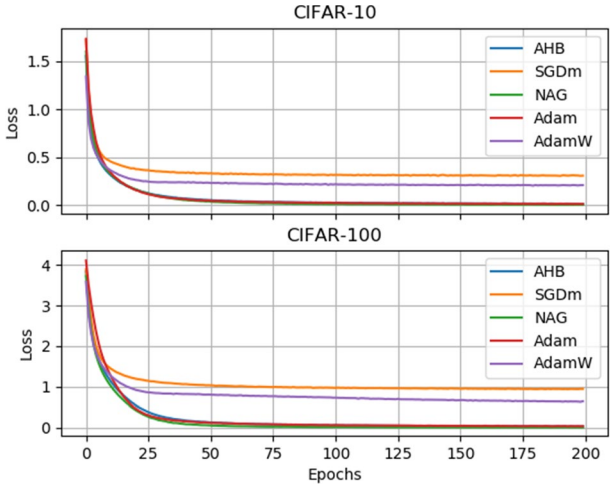

Fig. 8 The training loss at every epoch for all optimizers on both CIFAR-10 (top) and CIFAR-100 (bottom). The plots reflect the mean training loss over the 3 random runs

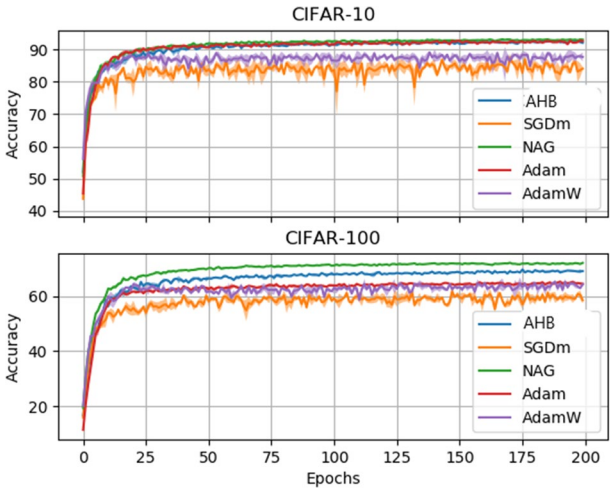

Fig. 9 The test set accuracy at every epoch for all optimizers on both CIFAR-10 (top) and CIFAR-100 (bottom). The solid lines reflect the mean accuracy over the 3 random runs, whereas the shaded regions reflect the corresponding added standard deviations

Table 4 Average test accuracy for CIFAR-10 (top) and CIFAR-100 (bottom) over last 10 epochs

AHB	SGDm	NAG	Adam	AdamW
92.35%	84.16%	**93.03%**	92.38%	87.64%
±0.25%	±1.44%	**± 0.17%**	±0.24%	±1.28%
69.04%	59.58%	**71.83%**	64.79%	63.95%
±0.37%	±1.16%	**± 0.24%**	±0.35%	±0.78%

Bold reflect the best performing metric values

We run SGD with a momentum factor of 0.9, learning rate of 0.05, and weight decay 0.001. AdamW is run with a learning rate of 3×10^{-4} and weight decay value of 1. For NAG, we use a learning rate of 0.05 and $\beta = 0.9$. Whereas for Adam, we choose a learning rate value of 0.005, with no weight decay and the standard values of $\beta_1 = 0.9$ and

$\beta_2 = 0.99$. Again, our method does not require the selection of a learning rate, as it automatically updates its hyper-parameters at every iteration.

Figures 8 and 9 show the mean training loss and the mean test set accuracy at every epoch over the three random runs for both datasets, respectively. The mean test accuracies over the last 10 epochs shown in Table 4. In CIFAR-10, our proposed optimizer achieves comparable test accuracy to the other optimizers, resulting in an average 92.35% mean accuracy over the last 10 epochs. Whereas for CIFAR-100, our proposed optimizer outperforms all the other optimizers, except NAG, by achieving a mean test accuracy of 69.04% over the last 10 epochs, where Adam achieves a mean test accuracy of 64.79% over the last 10 epochs. The best performing optimizer however for these tasks is the NAG optimizer. We point out however that with the use of learning-rate scheduling, where the learning-rate is decayed by some factor when the error stagnates, better performances can be achieved, such as 95% test accuracy on CIFAR-10 using SGDm (Lang et al., 2019). We reiterate that our main objective is tuning reduction, and not necessarily outperforming the other optimizers. Thus, for fair comparison, we compare our method with other optimizers while only tuning their hyper-parameters, and not with learning-rate scheduling implemented, which can be considered extensive. We set $\gamma = 1$ (no tuning) in all the image classification datasets yielding competitive convergence rates in regard to the popular first-order methods that *require excessive tuning of more than one hyper-parameter*.

6 Conclusion

In this paper, we have developed an adaptive HB method designed to tackle large-scale systems, which requires the tuning of only a single hyper-parameter, and in some cases requires no tuning at all. Our method approximates the optimal parameters set forth by Polyak (1987) in an iterative manner. We showed that a linear convergence rate can be attained for strongly convex objective functions. Our method's potential was demonstrated against several optimizers on a number of tasks, such as positive semi-definite quadratic functions, the strongly convex function introduced by Lessard et al. (2016) (with and without noise), the non-convex Beale function, and image classification tasks using deep neural networks. We find that our optimizer displays superior performance on many of these tasks, including the inherent capability to suppress gradient noise, and performs comparably well to popular optimizers on image classification tasks with no tuning performed whatsoever.

There are many possible interesting extensions of our work, such as developing a gain scheduling technique tailored to the design of our proposed optimizer. Such a scheduling scheme can be formulated in such a way that the parameter γ of Eq. (2) is gradually increased once the loss begins to saturate.

Appendix

This section is organized as follows. "Appendix A" provides useful preliminaries. We present a lemma, Lemma 1, in "Appendix B" where the proofs of Proposition 1 and Lemma 1 are included in "Appendix B". At the end of "Appendix B", we provide an illustrative example. We also present an additional lemma, Lemma 2, in "Appendix C" where the proofs of Theorem 1 and Lemma 2 are also included in "Appendix C".

Appendix A: Preliminaries

We first introduce some properties that hold for L-smooth functions, as defined in Definition 1, for all $x, y \in \mathbb{R}^d$ from Nesterov (2003):

$$f(y) - f(x) - \langle \nabla f(x), y - x \rangle \leq \frac{L}{2}||x - y||^2 \tag{15}$$

and:

$$\frac{1}{L}||\nabla f(x)||^2 \leq \langle x - x^*, \nabla f(x) \rangle \tag{16}$$

$$f(x) - f(x^*) + \frac{1}{2L}||\nabla f(x)||^2 \leq \langle x - x^*, \nabla f(x) \rangle \tag{17}$$

If $f(x)$ is a convex continuously differentiable function, then, we have

$$f(y) - f(x) \geq \langle y - x, \nabla f(x) \rangle. \tag{18}$$

In addition, if $f(x)$ is strongly convex, then from Theorem 2.1.10 (Nesterov, 2003) we have for all $x, y \in \mathbb{R}^d$

$$\frac{1}{2\mu}||\nabla f(y) - \nabla f(x)||^2 \geq f(y) - f(x) - \langle \nabla f(x), y - x \rangle \tag{19}$$

and

$$\langle \nabla f(y) - \nabla f(x), y - x \rangle \geq \frac{\mu}{2}||y - x||^2 \tag{20}$$

Appendix B: Proof of Proposition 1

Before we prove Proposition 1, we present a lemma, Lemma 1, and then simultaneously prove all items in Proposition 1 and Lemma 1.

Proposition 1 *Suppose that $f(x)$ is strongly convex and L-smooth. Then, $L \geq \mu$, and the algorithm (1) with hyper-parameters γ, α_k and β_k in (4), guarantees the following properties for all $k \geq 0$:*

(P1) $\gamma\mu \leq \hat{L}_k \leq \gamma L$.

For all $\gamma \geq \frac{L}{\mu} + \sqrt{\frac{L^2}{\mu^2} - 1}$, we have

(P2) $\sqrt{\gamma^2\mu^2 + \mu^2 - 2\gamma\mu L} \leq \hat{\mu}_k \leq (\gamma + 1)L$.

(P3) $\beta_k \in (0, 1)$, and

$$\frac{4}{(\sqrt{\gamma}+\sqrt{\gamma+1})^2 L} \leq \alpha_k \leq \frac{4}{(\sqrt{\gamma\mu}+(\gamma^2\mu^2+\mu^2-2\gamma\mu L)^{\frac{1}{4}})^2}.$$

(*P*4) $\hat{\mu}_k$ *is an increasing function of* γ, *and* α_k *is a decreasing function of* γ.

(*P*5) $\lim_{\gamma\to\infty} \hat{\mu}_k = \lim_{\gamma\to\infty} \hat{L}_k = \infty$, *and*

$\lim_{\gamma\to\infty} \alpha_k = \lim_{\gamma\to\infty} \beta_k = 0$.

Lemma 1 *Given any* $\epsilon_1 > 0$, *then for all* $k \geq 0$ *there exists a* $\gamma_1 > \frac{L}{\mu} + \sqrt{\frac{L^2}{\mu^2} - 1}$ *such that for all* $\gamma \geq \gamma_1$, $\frac{\beta_k^2}{\alpha_k} \leq \epsilon_1$.

Proof of Proposition 1 and *Lemma* 1. From (2), we can write $\hat{L}_k = \gamma\sqrt{P_k}$, where

$$P_k \triangleq \frac{h_k^T h_k}{\Delta x_k^T \Delta x_k}.$$

We first show (P1); that is, $\gamma\mu \leq \hat{L}_k \leq \gamma L$ for all k. Since $f(x)$ is L-smooth, then $||\nabla f(x_k) - \nabla f(x_{k-1})|| \leq L||x_k - x_{k-1}||$. Consequently, $h_k^T h_k = ||\nabla f(x_k) - \nabla f(x_{k-1})||^2 \leq L^2||\Delta x_k||^2$, where $h_k = \nabla f(x_k) - \nabla f(x_{k-1})$ and $\Delta x_k = x_k - x_{k-1}$. Thus, $P_k \leq L^2$ and for $\gamma \geq 1$, $\hat{L}_k \leq \gamma L$.

Using (19), we have

$$\frac{1}{2\mu} h_k^T h_k \geq f(x_{k-1}) - f(x_k) + \langle \nabla f(x_k), \Delta x_k \rangle. \tag{21}$$

From (5), we have

$$f(x_{k-1}) - f(x_k) \geq -\langle \Delta x_k, \nabla f(x_k) \rangle + \frac{\mu}{2}||\Delta x_k||^2$$

or

$$\langle \Delta x_k, \nabla f(x_k) \rangle \geq \frac{\mu}{2}||\Delta x_k||^2 - f(x_{k-1}) + f(x_k).$$

Inserting the last inequality into (21), we obtain

$$\frac{1}{2\mu} h_k^T h_k \geq \frac{\mu}{2}||\Delta x_k||^2.$$

Therefore, $h_k^T h_k \geq \mu^2||\Delta x_k||^2$.

Thus, $||\nabla f(x_k) - \nabla f(x_{k-1})|| \geq \mu||x_k - x_{k-1}||$ and we have $||\nabla f(x_k) - \nabla f(x_{k-1})|| \leq L||x_k - x_{k-1}||$; hence, $L \geq \mu$. $P_k \geq \mu^2$, and $\hat{L}_k \geq \gamma\mu$. Since $\mu > 0$, then for sufficiently large γ, $\hat{L}_k$ can be made arbitrarily large.

Next, we show (P2)-(P4). We already showed that $\gamma\mu \leq \hat{L}_k \leq \gamma L$. From (3), we define p_k such that $\hat{\mu}_k = \sqrt{p_k}$, where

$$p_k \triangleq \frac{(h_k - \hat{L}_k \Delta x_k)^T (h_k - \hat{L}_k \Delta x_k)}{\Delta x_k^T \Delta x_k}. \tag{22}$$

By expanding the quadratic expression of p_k, we get

$$p_k = \hat{L}_k^2 + \frac{\|h_k\|^2}{\|\Delta x_k\|^2} - 2\hat{L}_k \frac{h_k^T \Delta x_k}{\|\Delta x_k\|^2}.$$

We have already shown that $\mu^2 \leq \frac{\|h_k\|^2}{\|\Delta x_k\|^2} \leq L^2$, and from Cauchy-Schwarz inequality, we have $-h_k^T \Delta x_k \leq \|h_k\| \|\Delta x_k\|$ and $-h_k^T \Delta x_k \geq -\|h_k\| \|\Delta x_k\|$. Therefore,

$$p_k \leq \hat{L}_k^2 + L^2 + 2\hat{L}_k L = (\hat{L}_k + L)^2, \text{and}$$
$$p_k \geq \hat{L}_k^2 + \mu^2 - 2L\hat{L}_k.$$

Consequently,

$$\sqrt{\hat{L}_k^2 + \mu^2 - 2L\hat{L}_k} \leq \hat{\mu}_k \leq \hat{L}_k + L. \tag{23}$$

The lower bound of $\hat{\mu}_k$ requires that $\hat{L}_k^2 + \mu^2 - 2L\hat{L}_k \geq 0$. This inequality holds whenever $\hat{L}_k \geq L + \sqrt{L^2 - \mu^2}$. Since $\hat{L}_k \geq \gamma\mu$, then for $\gamma \geq \frac{L}{\mu} + \sqrt{\frac{L^2}{\mu^2} - 1}$, we have $\hat{L}_k^2 + \mu^2 - 2L\hat{L}_k \geq 0$, for all k.

From its definition, it is clear that $0 < \beta_k < 1$. To find the lower bound of α_k, we use the upper bounds of both $\hat{L}_k$ and $\hat{\mu}_k$. Since $\gamma\mu \leq \hat{L}_k \leq \gamma L$, then using (23), we obtain $\hat{\mu}_k \leq (\gamma + 1)L$, and

$$\alpha_k \geq \frac{4}{(\sqrt{\gamma} + \sqrt{\gamma + 1})^2 L}. \tag{24}$$

Next, we show that $\hat{\mu}_k$ is an increasing function of γ for $\gamma > \frac{L}{\mu}$. We consider the partial derivative of p_k with respect to $\hat{L}_k$, that is,

$$\frac{\partial p_k}{\partial \hat{L}_k} = 2\hat{L}_k - 2\frac{\langle \nabla f(x_k) - \nabla f(x_{k-1}), x_k - x_{k-1} \rangle}{\Delta x_k^T \Delta x_k}.$$

We use (15) twice, one time with $y \equiv x_k$ and $x \equiv x_{k-1}$ and another time with $x \equiv x_k$ and $y \equiv x_{k-1}$, we obtain

$$\begin{aligned} \langle \nabla f(x_{k-1}), x_k - x_{k-1} \rangle &\geq f(x_k) - f(x_{k-1}) \\ &\quad - \frac{L}{2} ||x_k - x_{k-1}||^2 \\ -\langle \nabla f(x_k), x_k - x_{k-1} \rangle &\geq f(x_{k-1}) - f(x_k) \\ &\quad - \frac{L}{2} ||x_k - x_{k-1}||^2. \end{aligned}$$

We sum the above inequalities, multiply by 2 and divide by $\Delta x_k^T \Delta x_k$, we get

$$-2\frac{\langle \nabla f(x_k) - \nabla f(x_{k-1}), x_k - x_{k-1} \rangle}{\Delta x_k^T \Delta x_k} \geq -2L.$$

Consequently,

$$\frac{\partial p_k}{\partial \hat{L}_k} \geq 2(\hat{L}_k - L).$$

Recall that $\hat{L}_k \geq \gamma\mu$. Thus, for $\gamma > \frac{L}{\mu}$, we can make $\hat{L}_k > L$, hence $\frac{\partial p_k}{\partial \hat{L}_k} > 0$. Thus, since $\hat{\mu}_k = \sqrt{p_k}$, then $\hat{\mu}_k$ becomes an increasing function of γ.

Next, we find the upper bound of α_k where we use the lower bounds of both $\hat{L}_k$ and $\hat{\mu}_k$. The lower bound of $\hat{\mu}_k$ holds whenever $\gamma \geq \frac{L}{\mu} + \sqrt{\frac{L^2}{\mu^2} - 1} \geq \frac{L}{\mu}$, where $\hat{\mu}_k$ is an increasing function of γ and note that from (23), we find that the lower bound is an increasing function of $\hat{L}_k$ for $\gamma > \frac{L}{\mu}$. Since $\hat{L}_k \geq \gamma\mu$, then $\hat{\mu}_k \geq \sqrt{\gamma^2\mu^2 + \mu^2 - 2\gamma\mu L}$. Thus, for $\gamma \geq \frac{L}{\mu} + \sqrt{\frac{L^2}{\mu^2} - 1}$, we obtain

$$\alpha_k \leq \frac{4}{(\sqrt{\gamma\mu} + (\gamma^2\mu^2 + \mu^2 - 2\gamma\mu L)^{\frac{1}{4}})^2}. \tag{25}$$

Next, we show that for $\gamma > \frac{L}{\mu}$, α_k is a decreasing function of γ. From the definition of α_k, it is readily clear that as γ increases, both $\hat{L}_k$ and $\hat{\mu}_k$ increase, and α_k decreases.

Next, we prove Lemma 1. From definitions of α_k and β_k, we have

$$\frac{\beta_k^2}{\alpha_k} = \frac{\left(\sqrt{\hat{L}_k} - \sqrt{\hat{\mu}_k}\right)^4}{4\left(\sqrt{\hat{L}_k} + \sqrt{\hat{\mu}_k}\right)^2}. \tag{26}$$

The inequality in (23) implies that

$$\hat{L}_k\sqrt{1 + \frac{\mu^2}{\hat{L}_k^2} - \frac{2L}{\hat{L}_k}} \leq \hat{\mu}_k \leq \hat{L}_k\Big(1 + \frac{L}{\hat{L}_k}\Big),$$

and we know that $\mu\gamma \leq \hat{L}_k \leq L\gamma$. Thus, as $\gamma \to \infty$, $\hat{L}_k \to \infty$. Notice that

$$\begin{aligned}\sqrt{\hat{L}_k}\Big(1 - \sqrt{1 + \frac{L}{\hat{L}_k}}\Big) &\leq \sqrt{\hat{L}_k} - \sqrt{\hat{\mu}_k} \\ &\leq \sqrt{\hat{L}_k}\Big(1 - \Big(1 + \frac{\mu^2}{\hat{L}_k^2} - \frac{2L}{\hat{L}_k}\Big)^{\frac{1}{4}}\Big),\end{aligned}$$

and

$$1 - \sqrt{1 + \frac{L}{\hat{L}_k}} \leq 1 - \sqrt{\frac{\hat{\mu}_k}{\hat{L}_k}} \quad \leq 1 - \Big(1 + \frac{\mu^2}{\hat{L}_k^2} - \frac{2L}{\hat{L}_k}\Big)^{\frac{1}{4}}.$$

Since $0 < \mu\gamma \leq \hat{L}_k \leq L\gamma$ for all k, we have
$\frac{L}{\mu\gamma} \geq \frac{L}{\hat{L}_k}$ or $1 - \sqrt{1 + \frac{L}{\mu\gamma}} \leq 1 - \sqrt{1 + \frac{L}{\hat{L}_k}}$.

In addition, $\frac{\mu^2}{\hat{L}_k^2} \geq \frac{\mu^2}{L^2\gamma^2}$ and $-\frac{2L}{\hat{L}_k} \geq -\frac{2L}{\mu\gamma}$. Then, $1 + \frac{\mu^2}{\hat{L}_k^2} - \frac{2L}{\hat{L}_k} \geq 1 + \frac{\mu^2}{L^2\gamma^2} - \frac{2L}{\hat{L}_k} \geq 1 + \frac{\mu^2}{L^2\gamma^2} - \frac{2L}{\mu\gamma}$. Thus, $1 - \Big(1 + \frac{\mu^2}{\hat{L}_k^2} - \frac{2L}{\hat{L}_k}\Big)^{\frac{1}{4}} \leq 1 - \Big(1 + \frac{\mu^2}{L^2\gamma^2} - \frac{2L}{\mu\gamma}\Big)^{\frac{1}{4}}$.

Consequently,

$$1-\sqrt{1+\frac{L}{\mu\gamma}} \leq 1-\sqrt{\frac{\hat{\mu}_k}{\hat{L}_k}} \leq 1-\left(1+\frac{\mu^2}{L^2\gamma^2}-\frac{2L}{\mu\gamma}\right)^{\frac{1}{4}}.$$

Therefore, as $\gamma \to \infty$, both upper and lower bounds of $1-\sqrt{\frac{\hat{\mu}_k}{\hat{L}_k}}$ approach zero; hence $\lim_{\gamma\to\infty} \frac{\hat{\mu}_k}{\hat{L}_k} = 1$ for all $k \geq 0$ since the bounds are independent of k.

Consequently, as $\gamma \to \infty$, the numerator in (26) remains bounded, whereas the denominator would tend to infinity. Since $\alpha_k > 0, \forall k$, then as $\gamma \to \infty$, $\frac{\beta_k^2}{\alpha_k} \to 0$ for all $k \geq 0$. Therefore, given any $\epsilon_1 > 0$, then for all $k \geq 0$ there exists a $\gamma_1 > \frac{L}{\mu} + \sqrt{\frac{L^2}{\mu^2}-1}$ such that for all $\gamma \geq \gamma_1$, $\frac{\beta_k^2}{\alpha_k} \leq \epsilon_1$.

Finally, we show (P5). We showed that as $\gamma \to \infty$, $\hat{L}_k \to \infty$, and $\lim_{\gamma\to\infty} \hat{\mu}_k = \infty$. It is readily clear that as $\gamma \to \infty$, $\alpha_k \to 0$ for all $k \geq 0$. Using the same argument of the proof of Lemma 1 whereas $\gamma \to \infty$, the numerator of β_k remains bounded and its denominator goes to infinity. Thus, as $\gamma \to \infty$, $\beta_k \to 0$.

Example This example is intended just to show that (2) and (3) *can* immediately estimate the largest eigenvalue and the smallest non-zero eigenvalue of A.

We consider the following matrix

$$A = \begin{bmatrix} 41 & 1 \\ 1 & 100 \end{bmatrix}$$

with eigenvalues $\{100.02, 40.98\}$. We use $\Delta x = [1\ 1]^T$. We adopt a tuned $\gamma = 1.3$ to obtain $\hat{L}_0 = 100.5$ and $\hat{\mu}_0 = 41.4$. However, with $\gamma = 2$, we obtain $\hat{L}_0 = 154.7$ and $\hat{\mu}_0 = 88.3$. As expected for larger values of γ, we obtain larger values of $\hat{L}$, which can exceed the maximum eigenvalue of A. □

Appendix C: Proof of Theorem 1

In what follows we modify Lemma 1 and Theorem 4 in Ghadimi et al. (2015), where the HB hyper- parameters are assumed constant, to our time-varying case (1).

Lemma 2 *Let $\{U_k\}_{k\geq 0}$ and $\{V_k\}_{k\geq 0}$ be non-negative sequences of real scalars satisfying*

$$U_{k+1} + v_{1,k+1}V_{k+1} \leq u_{1,k}U_k + u_{2,k}U_{k-1} + v_{2,k}V_k \tag{27}$$

where $\{v_{1,k}\}_{k\geq 0}$, $\{u_{1,k}\}_{k\geq 0}$ and $\{v_{2,k}\}_{k\geq 0}$ are positive sequences of real scalars, and $\{u_{2,k}\}_{k\geq 0}$ is a sequence of non-negative scalars. If $qu_{1,k} + u_{2,k} \leq q^2$ and $0 \leq \frac{v_{2,k}}{v_{1,k}} \leq q$, for $q \in (0, 1)$ and for all $k \geq 0$, then the sequence $\{U_k\}_{k\geq 0}$ generated by (27) satisfies

$$U_k \leq q^k((q + 1 - u_{1,0})U_0 + v_{1,0}V_0). \tag{28}$$

Proof of Lemma 2 It is readily clear that (28) holds for $k = 0$. We add zU_k, with $z \geq 0$ to both sides of (27)

$$U_{k+1} + zU_k + v_{1,k+1}V_{k+1} \leq (u_{1,k} + z)U_k + u_{2,k}U_{k-1} + v_{2,k}V_k \tag{29}$$

$$= (u_{1,k} + z)\Big(U_k + \frac{u_{2,k}}{u_{1,k} + z}U_{k-1} + \frac{v_{2,k}}{u_{1,k} + z}V_k\Big) \tag{30}$$

if $U_0 = U_{-1}$ and the following two inequalities hold

$$\frac{u_{2,k}}{u_{1,k} + z} \leq z \tag{31}$$

$$\frac{v_{2,k}}{u_{1,k} + z} \leq v_{1,k}, \tag{32}$$

then $S_{k+1} \leq (z + u_{1,k})S_k$, where $S_k \triangleq U_k + zU_{k-1} + v_{1,k}V_k$. For (31) to hold, we need $z^2 + u_{1,k}z - u_{2,k} \geq 0, \forall k$ or $z \geq \frac{-u_{1,k}+\sqrt{u_{1,k}^2+4u_{2,k}}}{2}, \forall k$. In addition, for (32) to hold, we need $z \geq \frac{v_{2,k}}{v_{1,k}} - u_{1,k}, \forall k$. Therefore, for both (31) and (32) to hold, we need

$$z \geq \max\left\{\frac{-u_{1,k} + \sqrt{u_{1,k}^2 + 4u_{2,k}}}{2}, \frac{v_{2,k}}{v_{1,k}} - u_{1,k}\right\}, \forall k. \tag{33}$$

In order to have S_k converges to zero at a linear rate, it is sufficient to have $0 \leq z + u_{1,k} \leq q < 1, \forall k$, or

$$0 \leq \max\left\{\frac{u_{1,k} + \sqrt{u_{1,k}^2 + 4u_{2,k}}}{2}, \frac{v_{2,k}}{v_{1,k}}\right\} \leq q < 1, \forall k. \tag{34}$$

Note that since $u_{1,k} \geq 0$ and $u_{2,k} \geq 0$, then the first element of $\max\{.,.\}$ is non-negative. In addition, for $\frac{u_{1,k}+\sqrt{u_{1,k}^2+4u_{2,k}}}{2} \leq q$, then we need $u_{1,k}^2 + 4u_{2,k} \leq (2q - u_{1,k})^2$ or $qu_{1,k} + u_{2,k} \leq q^2$. Furthermore, $0 \leq \frac{v_{2,k}}{v_{1,k}} \leq q$.

Since $0 \leq z + u_{1,k} \leq q < 1, \forall k$, then

$$\begin{aligned} U_{k+1} + zU_k + v_{1,k+1}V_{k+1} &\leq q(U_{k+1} + zU_{k-1} + v_{1,k}V_k) \\ &\leq \cdots \leq q^{k+1}((1 + z)U_0 + v_{1,0}V_0) \\ &\leq q^{k+1}((q + 1 - u_{1,0})U_0 + v_{1,0}V_0) \end{aligned}$$

Since $S_k = U_k + zU_{k-1} + v_{1,k}V_k$ is the summation of non-negative scalars, this ends the proof. □

Theorem 1 *Suppose that $f(x)$ is strongly convex and L-smooth. Then, there exists a $\gamma_0 \geq \max\left\{4\left(\frac{L}{\mu} + \sqrt{\frac{L^2}{\mu^2} - 1}\right), \frac{4(L+1)}{\mu}\right\}$ such for $\gamma > \gamma_0$, the sequence $\{x_k\}$ generated by algorithm (1) with step-size α_k and hyper-parameter β_k in (4), satisfies*

$$f(x_k) - f(x^*) \leq q^k(f(x_0) - f(x^*)) \tag{35}$$

where $q \in (0, 1)$, and x^* is the unique optimal solution that minimizes $f(x)$.

Proof of Theorem 1 Since $f(x)$ is L-smooth, then

$$\begin{aligned} f(x_{k+1}) - f(x_k) \leq & \langle \nabla f(x_k), x_{k+1} - x_k \rangle \\ & + \frac{L}{2} ||x_{k+1} - x_k||^2 \end{aligned} \tag{36}$$

We have from (1),

$$\begin{aligned} ||x_{k+1} - x_k||^2 = & \alpha_k^2 ||\nabla f(x_k)||^2 + \beta_k^2 ||x_k - x_{k-1}||^2 \\ & - 2\alpha_k \beta_k \langle \nabla f(x_k), x_k - x_{k-1} \rangle \end{aligned} \tag{37}$$

and

$$\begin{aligned} \langle \nabla f(x_k), x_{k+1} - x_k \rangle = & -\alpha_k ||\nabla f(x_k)||^2 \\ & + \beta_k \langle \nabla f(x_k), x_k - x_{k-1} \rangle \end{aligned}$$

Substituting the above equalities into (36), subtracting $f(x^*)$ from each side and arranging terms, we obtain

$$\begin{aligned} & f(x_{k+1}) - f(x^*) \leq f(x_k) - f(x^*) + \frac{L\beta_k^2}{2} ||x_k - x_{k-1}||^2 \\ & - \alpha_k \left(1 - \frac{\alpha_k L}{2}\right) ||\nabla f(x_k)||^2 \\ & + \beta_k (1 - L\alpha_k) \langle \nabla f(x_k), x_k - x_{k-1} \rangle \end{aligned} \tag{38}$$

We multiply both sides of (37) by $\frac{1-\alpha_k L}{2\alpha_k}$, add the resulting identity to (38), and rearrange terms yields

$$\begin{aligned} & f(x_{k+1}) - f(x^*) + \frac{1 - \alpha_k L}{2\alpha_k} ||x_{k+1} - x_k||^2 \leq f(x_k) - f(x^*) \\ & - \frac{\alpha_k}{2} ||\nabla f(x_k)||^2 + \frac{\beta_k^2}{2\alpha_k} ||x_k - x_{k-1}||^2 \end{aligned} \tag{39}$$

Since $f(x)$ is smooth and μ-strongly convex, then from Nesterov (2003) or Polyak- Lojasiewicz (PL) inequality, we have

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

Thus, from (39) and the above inequality

$$\begin{aligned} & f(x_{k+1}) - f(x^*) + \frac{1 - \alpha_k L}{2\alpha_k} ||x_{k+1} - x_k||^2 \leq f(x_k) - f(x^*) \\ & - \mu\alpha_k (f(x_k) - f(x^*)) + \frac{\beta_k^2}{2\alpha_k} ||x_k - x_{k-1}||^2 \end{aligned} \tag{40}$$

We arrange the terms of (40) and put it in the form of (27), we obtain

$$f(x_{k+1}) - f(x^*) + v_{1,k+1}||x_{k+1} - x_k||^2 \\ \leq u_{1,k}(f(x_k) - f(x^*)) + u_{2,k}(f(x_{k-1}) - f(x^*)) \\ + v_{2,k}||x_k - x_{k-1}||^2$$

This inequality is in the form of (27) (Lemma 2) where we identify $f(x_k) - f(x^*)$ with U_k, and $||x_k - x_{k-1}||^2$ with V_k, and

$$v_{1,k+1} = \frac{1 - \alpha_k L}{2\alpha_k}, u_{1,k} = 1 - \alpha_k \mu, u_{2,k} = 0, v_{2,k} = \frac{\beta_k^2}{2\alpha_k}.$$

In what follows, we show that all conditions of Lemma 2 are satisfied.

From the definition of α_k, it is clear that $\alpha_k \leq \frac{4}{\hat{L}_k}$. Since $\hat{L}_k \geq \mu\gamma$, then $\alpha_k \leq \frac{4}{\mu\gamma}$ for all $k \geq 0$. Thus, for $\gamma > 4\left(\frac{L}{\mu} + \sqrt{\frac{L^2}{\mu^2} - 1}\right) \geq 4\frac{L}{\mu} \geq 4$, we have $0 < \alpha_k\mu < 1$, and $0 < 1 - \alpha_k\mu < 1$. Similarly, $\alpha_k L \leq \frac{4L}{\mu\gamma}$ and since $\gamma > 4\frac{L}{\mu}$, then $0 < \alpha_k L < 1$ and $1 - \alpha_k L > 0$. Thus, for all $k \geq 0$, $\{v_{1,k}\}_{k\geq 0}$ and $\{u_{1,k}\}_{k\geq 0}$ become sequences of positive scalars. It is readily clear that $\{v_{2,k}\}_{k\geq 0}$ is a sequence of positive scalars and $\{u_{2,k}\}_{k\geq 0}$ is a sequence of zero (or non-negative) scalars.

We are left to show that for some constant $q \in (0, 1)$, we have $qu_{1,k} + u_{2,k} \leq q^2$, and $0 \leq \frac{v_{2,k}}{v_{1,k}} \leq q$. We next consider, $qu_{1,k} + u_{2,k} = qu_{1,k} = q(1 - \alpha_k\mu)$.

Let $q = \frac{(\gamma+1)L-\mu}{(\gamma+1)L}$; hence, $q \in (0, 1)$. From Proposition 1 (P3), we have for all $k \geq 0$

$$\alpha_k \geq \frac{4}{(\sqrt{\gamma} + \sqrt{\gamma+1})^2 L} > \frac{1}{(\gamma+1)L}.$$

Consequently,

$$1 - \alpha_k\mu < 1 - \frac{\mu}{(\gamma+1)L} = q.$$

Therefore, $q(1 - \alpha_k\mu) < q^2$.

Next, we show $0 \leq \frac{v_{2,k}}{v_{1,k}} \leq q$ or $0 \leq \frac{\beta_k^2}{\alpha_k}\frac{\alpha_{k-1}}{1-\alpha_{k-1}L} \leq q$.

We first show that for $\gamma > \gamma_2 \triangleq \frac{4(L+1)}{\mu}$, we have $0 < \frac{\alpha_k}{1-\alpha_k L} < 1$ for all k. We know $\alpha_k > 0$ and we already showed that for $\gamma > 4\frac{L}{\mu}$, then $1 - \alpha_k L > 0$. Next, we consider

$$\frac{\alpha_k}{1 - \alpha_k L} = \frac{4}{\left(\sqrt{\hat{L}_k} + \sqrt{\hat{\mu}_k}\right)^2 - 4L} < \frac{4}{\gamma\mu - 4L}.$$

It is clear that the upper bound of $\frac{\alpha_k}{1-\alpha_k L}$ is less than one for $\gamma > \frac{4(L+1)}{\mu}$.

Thus, for all $k \geq 0$, $\frac{\beta_k^2}{\alpha_k}\frac{\alpha_{k-1}}{1-\alpha_{k-1}L} \leq \frac{\beta_k^2}{\alpha_k}$. From Lemma 1 we have for all $k \geq 0$ and for any $q > 0$, there exists a $\gamma_1 > \frac{L}{\mu} + \sqrt{\frac{L^2}{\mu^2} - 1} \geq \frac{L}{\mu}$ such that for all $\gamma \geq \gamma_1$, $\frac{\beta_k^2}{\alpha_k} \leq q$ or $\frac{v_{2,k}}{v_{1,k}} \leq q < 1$.

Therefore, for all for all $k \geq 0$ and $\gamma > \gamma_0 \triangleq \max\left\{\gamma_1, 4\left(\frac{L}{\mu} + \sqrt{\frac{L^2}{\mu^2} - 1}\right), \frac{4(L+1)}{\mu}\right\}$, we have all conditions of Lemma 2 satisfied. Therefore, one can apply Lemma 2 to conclude the linear convergence in (35). □

Appendix D: Proof of Theorem 2

We first state Theorem 1 in Gitman et al. (2019).

Theorem 1 *in Gitman et al.* (2019). *Let F satisfy Assumptions* 1–3. *Additionally, assume* $0 \leq v_k \leq 1$ *and the non-negative sequences* $\{\alpha_k\}$ *and* $\{\beta_k\}$ *satisfy the following conditions*:

$$\sum_{k=0}^{\infty} \alpha_k = \infty, \sum_{k=0}^{\infty} \alpha_k^2 < \infty, \lim_{k\to\infty} \beta_k = 0, \bar{\beta} \triangleq \sup_k \beta_k < 1. \tag{41}$$

Then the sequence $\{x_k\}$ generated by the QHM algorithm satisfies

$$\liminf_{k\to\infty} \|\nabla F(x_k)\| = 0 \quad a.s.$$

Moreover, we have

$$\limsup_{k\to\infty} F(x_k) = \limsup_{k\to\infty, \|\nabla F(x_k)\|\to 0} F(x_k) \quad a.s.$$

The QHM algorithm with $v_k = 1$ is the Stochastic Heavy Ball method.

Proof of Theorem 2 The hyper-parameters of our SAHB are given by: $\hat{\alpha}_k = \min\left\{\alpha_k, \frac{C}{(k+1)^\nu}\right\}$, $\hat{\beta}_k = \min\left\{\beta_k, \frac{1}{(k+2)^\nu}\right\}$, $0.5 < \nu \leq 1$, $C > 0$, and α_k and β_k are positive and defined in (4).

Making use of Theorem 1 in Gitman et al. (2019), we only need to show that

$$\sum_{k=0}^{\infty} \hat{\alpha}_k = \infty, \sum_{k=0}^{\infty} \hat{\alpha}_k^2 < \infty, \lim_{k\to\infty} \hat{\beta}_k = 0, \bar{\beta} \triangleq \sup_k \hat{\beta}_k < 1.$$

Since $\hat{\beta}_k \leq \frac{1}{(k+2)^\nu}$, then $\sup_k \hat{\beta}_k < 1$ and $\lim_{k\to\infty} \hat{\beta}_k = 0$. In addition, since $\hat{\alpha}_k \leq \frac{C}{(k+1)^\nu}$ and $0.5 < \nu \leq 1$, then by using the integral test we have $\sum_{k=0}^{\infty} \hat{\alpha}_k^2 < \infty$.

Due to Lipschitz continuity of $\hat{h}$ (Assumption 4), we obtain $\|\hat{h}_k\|^2 \leq L_G^2 \|\Delta x_k\|^2$. Therefore, $P_k = \frac{\|\hat{h}_k\|^2}{\|\Delta x_k\|^2} \leq L_G^2$, which implies that $\hat{L}_k \leq \gamma L_G$.

On the other hand, $\hat{\mu}_k \triangleq \frac{\|\hat{h}_k - \hat{L}_k \Delta x_k\|}{\|\Delta x_k\|} \leq \frac{\|\hat{h}_k\|}{\|\Delta x_k\|} + \hat{L}_k$, thus, $\hat{\mu}_k \leq (\gamma+1) L_G$. Since $\alpha_k \triangleq \frac{4}{(\sqrt{\hat{L}_k} + \sqrt{\hat{\mu}_k})^2}$, we conclude that

$$\alpha_k \geq \frac{4}{(\sqrt{\gamma} + \sqrt{\gamma+1})^2 L_G}.$$

Hence, there is a finite K such that $\hat{\alpha}_k = C/(k+1)^\nu$ for all $k \geq K$. Since $\hat{\alpha}_k \geq 0$, $\sum_{k=0}^{\infty} \hat{\alpha}_k \geq \sum_{k=K}^{\infty} \hat{\alpha}_k = \infty$.

Since all the sufficient conditions in Gitman et al. (2019) are satisfied, then we deduce both (11) and (12), and this ends the proof. □

Author contributions S. Saab Jr has conceptualized the study including the theoretical work, performed the experiments, and written the manuscript. S. Phoha and A. Ray have supervised the course of the article, M. Zhu has contributed to the theoretical work, and supervised and organized the course of the article.

Funding The first author has been supported by the Walker Fellowship from the Applied Research Laboratory at the Pennsylvania State University. The work reported here has been supported in part by NSF CAREER award ECCS-1846706, and by the U.S. Air Force Office of Scientific Research (AFOSR) under Grant No. FA9550-15-1-0400 in the area of Dynamic Data-Driven Application Systems (DDDAS). Any opinions, findings and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the sponsoring agencies.

Availability of data and materials The data is based on open-source repository.

Code availability The codes will be made available upon acceptance of the manuscript.

Declarations

Conflict of interest Not Applicable (no potential conflict of interest is reported by the authors).

Ethics approval Not Applicable (the study does not involve human subjects and/or animals).

Consent to participate All the authors mentioned in the manuscript have agreed for authorship, read and approved the manuscript, and given consent for submission.

Consent for publication All the authors mentioned in the manuscript have given consent for submission and subsequent publication of the manuscript. The manuscript will not be submitted elsewhere until the editorial process is completed.

References

Bakirov, R., Fay, D., & Gabrys, B. (2021). Automated adaptation strategies for stream learning. *Machine Learning*, 1–34.

Beck, A. (2017). *First-order methods in optimization*. SIAM.

Berrada, L., Zisserman, A., & Kumar, M. P. (2020). Training neural networks for and by interpolation. In *International conference on machine learning* (pp. 799–809).

Bertsekas, D. P. (1997). A new class of incremental gradient methods for least squares problems. *SIAM Journal on Optimization, 7*(4), 913–926.

Carmon, Y., Duchi, J. C., Hinder, O., & Sidford, A. (2017). "convex until proven guilty": Dimensionfree acceleration of gradient descent on non-convex functions. In *International conference on machine learning* (pp. 654–663).

Cutkosky, A., & Mehta, H. (2020). Momentum improves normalized SGD. In *International conference on machine learning* (pp. 2260–2268).

Dauphin, Y. N., De Vries, H., & Bengio, Y. (2015). Equilibrated adaptive learning rates for non-convex optimization. arXiv preprint arXiv:1502.04390.

Dozat, T. (2016). Incorporating nesterov momentum into adam. In *ICLR Workshop*

Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, *12*(7).

Floridi, L., & Chiriatti, M. (2020). Gpt-3: Its nature, scope, limits, and consequences. *Minds and Machines, 30*(4), 681–694.

Gadat, S., Panloup, F., & Saadane, S. (2018). Stochastic heavy ball. *Electronic Journal of Statistics, 12*(1), 461–529.

Gaivoronski, A. A. (1994). Convergence properties of backpropagation for neural nets via theory of stochastic gradient methods. Part 1. *Optimization Methods and Software, 4*(2), 117–134.

Ghadimi, E. (2015). Accelerating convergence of largescale optimization algorithms (Unpublished doctoral dissertation). KTH Royal Institute of Technology.

Ghadimi, E., Feyzmahdavian, H. R., & Johansson, M. (2015). Global convergence of the heavy-ball method for convex optimization. In *2015 European control conference* (pp. 310–315).

Ghadimi, S., & Lan, G. (2013). Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization, 23*(4), 2341–2368.

Ghadimi, S., & Lan, G. (2016). Accelerated gradient methods for nonconvex nonlinear and stochastic programming. *Mathematical Programming, 156*(1–2), 59–99.
Gitman, I., Lang, H., Zhang, P., & Xiao, L. (2019). Understanding the role of momentum in stochastic gradient methods. *Advances in Neural Information Processing Systems*, *32*.
Gower, R. M., Loizou, N., Qian, X., Sailanbayev, A., Shulgin, E., & Richtárik, P. (2019). SGD: General analysis and improved rates. In *International conference on machine learning* (pp. 5200–5209).
He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778).
Hendrikx, H., Xiao, L., Bubeck, S., Bach, F., & Massoulie, L. (2020). Statistically preconditioned accelerated gradient method for distributed optimization. In *International conference on machine learning* (pp. 4203–4227).
Hinton, G., Srivastava, N., & Swersky, K. (2012). Lecture 6d-a separate, adaptive learning rate for each connection. *Slides of Lecture Neural Networks for Machine Learning, 1,* 1–31.
Hu, C., Kwok, J. T.-Y., & Pan, W. (2009). Accelerated gradient methods for stochastic optimization and online learning. In *Proceedings of the 23rd annual conference on neural information processing systems*.
Huang, F., Gao, S., Pei, J., & Huang, H. (2020). Momentum-based policy gradient methods. In *International conference on machine learning* (pp. 4422–4433).
Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4700–4708).
Jamil, M., & Yang, X.-S. (2013). A literature survey of benchmark functions for global optimisation problems. *International Journal of Mathematical Modelling and Numerical Optimisation, 4*(2), 150–194.
Khan, M., Nielsen, D., Tangkaratt, V., Lin, W., Gal, Y., & Srivastava, A. (2018). Fast and scalable bayesian deep learning by weight-perturbation in adam. In *International conference on machine learning* (pp. 2611–2620).
Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.
Koehler, G. (2020). MNIST handwritten digit recognition in pytorch. Retrieved from https://nextjournal.com/gkoehler/pytorch-mnist
Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images. *Technical report, University of Toronto*.
Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems, 25,* 1097–1105.
Lang, H., Zhang, P., & Xiao, L. (2019). Using statistics to automate stochastic optimization. arXiv preprint arXiv:1909.09785.
Lessard, L., Recht, B., & Packard, A. (2016). Analysis and design of optimization algorithms via integral quadratic constraints. *SIAM Journal on Optimization, 26*(1), 57–95.
Li, X., & Orabona, F. (2019). On the convergence of stochastic gradient descent with adaptive stepsizes. In *The 22nd international conference on artificial intelligence and statistics* (pp. 983–992).
Lin, Q., Lu, Z., & Xiao, L. (2015). An accelerated randomized proximal coordinate gradient method and its application to regularized empirical risk minimization. *SIAM Journal on Optimization, 25*(4), 2244–2273.
Liu, L., Jiang, H., He, P., Chen, W., Liu, X., Gao, J., & Han, J. (2019). On the variance of the adaptive learning rate and beyond. arXiv preprint arXiv:1908.03265 .
Loizou, N., & Richtárik, P. (2020). Momentum and stochastic momentum for stochastic gradient, newton, proximal point and subspace descent methods. *Computational Optimization and Applications, 77*(3), 653–710.
Loizou, N., Vaswani, S., Laradji, I., & Lacoste-Julien, S. (2020). Stochastic Polyak step-size for SGD: An adaptive learning rate for fast convergence. arXiv preprint arXiv:2002.10542.
Loshchilov, I., & Hutter, F. (2017). Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101.
Mai, V., & Johansson, M. (2020). Convergence of a stochastic gradient method with momentum for nonsmooth non-convex optimization. In *International conference on machine learning* (pp. 6630–6639).
Mises, R., & Pollaczek-Geiringer, H. (1929). Praktische verfahren der gleichungsauflösung. *ZAMMJournal of Applied Mathematics and Mechanics/ Zeitschrift für Angewandte Mathematik und Mechanik, 9*(1), 58–77.
Nesterov, Y. (1983). A method of solving a convex programming problem with convergence rate O(1=k2). In Sov. math. dokl (Vol. 27).
Nesterov, Y. (2003). *Introductory lectures on convex optimization: A basic course* (Vol. 87). Springer.

Oberman, A. M., & Prazeres, M. (2019). Stochastic gradient descent with Polyak's learning rate. arXiv preprint arXiv:1903.08688.
Ochs, P., Brox, T., & Pock, T. (2015). ipiasco: inertial proximal algorithm for strongly convex optimization. *Journal of Mathematical Imaging and Vision, 53*(2), 171–181.
Ochs, P., Chen, Y., Brox, T., & Pock, T. (2014). ipiano: Inertial proximal algorithm for nonconvex optimization. *SIAM Journal on Imaging Sciences, 7*(2), 1388–1419.
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, B. T. (1987). Introduction to optimization. Translations series in mathematics and engineering. Optimization Software.
Probst, P., Boulesteix, A.-L., & Bischl, B. (2019). Tunability: Importance of hyperparameters of machine learning algorithms. *Journal of Machine Learning Research, 20*(53), 1–32.
Qian, N. (1999). On the momentum term in gradient descent learning algorithms. *Neural Networks, 12*(1), 145–151.
Reddi, S. J., Kale, S., & Kumar, S. (2019). On the convergence of adam and beyond. arXiv preprint arXiv:1904.09237 .
Robbins, H., & Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics*, 400–407.
Rolinek, M., & Martius, G. (2018). L4: Practical lossbased stepsize adaptation for deep learning. arXiv preprint arXiv:1802.05074 .
Scieur, D., & Pedregosa, F. (2020). Universal asymptotic optimality of Polyak momentum. In *International conference on machine learning* (pp. 8565–8572).
Simsekli, U., Zhu, L., Teh, Y. W., & Gurbuzbalaban, M. (2020). Fractional underdamped langevin dynamics: Retargeting SGD with momentum under heavy-tailed gradient noise. In *International conference on machine learning* (pp. 8970–8980).
Song, H., Kim, S., Kim, M., & Lee, J.-G. (2020). Ada-boundary: accelerating DNN training via adaptive boundary batch selection. *Machine Learning, 109*(9), 1837–1853.
Tao, W., Long, S., Wu, G., & Tao, Q. (2021). The role of momentum parameters in the optimal convergence of adaptive Polyak's heavy-ball methods. arXiv preprint arXiv:2102.07314 .
Van Scoy, B., Freeman, R. A., & Lynch, K. M. (2017). The fastest known globally convergent first-order method for minimizing strongly convex functions. *IEEE Control Systems Letters, 2*(1), 49–54.
Ward, R., Wu, X., & Bottou, L. (2019). Adagrad stepsizes: Sharp convergence over nonconvex landscapes. In *International conference on machine learning* (pp. 6677–6686).
Wilson, A. C., Roelofs, R., Stern, M., Srebro, N., & Recht, B. (2017). The marginal value of adaptive gradient methods in machine learning. arXiv preprint arXiv:1705.08292 .
Yu, Y., Xu, P., & Gu, Q. (2018). Third-order smoothness helps: Faster stochastic optimization algorithms for finding local minima. In *Advances in neural information processing systems*, 31 .
Zavriev, S., & Kostyuk, F. (1993). Heavy-ball method in nonconvex optimization problems. *Computational Mathematics and Modeling, 4*(4), 336–341.
Zeiler, M. D. (2012). Adadelta: an adaptive learning rate method. arXiv preprint arXiv:1212.5701.
Zhang, M., Lucas, J., Ba, J., & Hinton, G. E. (2019). Lookahead optimizer: k steps forward, 1 step back. In *Advances in neural information processing systems* (pp. 9597–9608).
Zou, F., Shen, L., Jie, Z., Zhang, W., & Liu, W. (2019). A sufficient condition for convergences of adam and rmsprop. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 11127–11135).