JOTA manuscript No.
(will be inserted by the editor)

NAG-GS: Semi-Implicit, Accelerated and Robust Stochastic Optimizer

Valentin Leplat · Daniil Merkulov · Aleksandr Katrutsa · Daniel Bershatsky · Olga Tsymboi · Ivan Oseledets

Received: date / Accepted: date

Abstract Stochastic Gradient Descent (SGD) and its momentum-based variants remain the foundation of large-scale machine learning optimization, especially for training deep neural networks. These methods can often be interpreted as discretizations of underlying stochastic differential equations (SDEs). In this paper, we propose a novel and robust accelerated optimizer, called *NAG-GS*, derived from a second-order Nesterov-inspired SDE and discretized using a semi-implicit Gauss–Seidel scheme. We perform a detailed theoretical analysis of NAG-GS in the quadratic case, establishing convergence guarantees and deriving optimal learning rates that maximize the contraction rate while ensuring numerical stability. This is achieved through a careful spectral analysis of the iteration and stationary covariance matrices. Beyond theory, we demonstrate that NAG-GS is competitive with state-of-the-art optimizers such as Momentum SGD and AdamW across a wide range of tasks. These include logistic

Valentin Leplat, Corresponding author
Innopolis University
Innopolis, Russia
v.leplat@innopolis.ru

Daniil Merkulov
Skoltech, Moscow Institute of Physics and Technology
Moscow, Russia

Aleksandr Katrutsa
Skoltech, AIRI
Moscow, Russia

Daniel Bershatsky
Skoltech
Moscow, Russia

Olga Tsymboi
Moscow Institute of Physics and Technology, Sber AI Lab
Moscow, Russia

Ivan Oseledets
AIRI, Skoltech
Moscow, Russia

regression, deep residual networks on standard computer vision benchmarks, Transformers on the GLUE benchmark, and Vision Transformers. The method consistently exhibits strong empirical performance and robustness to large learning rates.

Keywords accelerated gradient method · Gauss-Seidel discretization · deep learning · stochastic first-order method

Mathematics Subject Classification (2000) 90C25 · 90C15 · 90C90

1 Introduction

We consider the problem of minimizing a smooth objective function based on stochastic gradient information:

$$\min_{x \in \mathbb{R}^n} f(x), \tag{1}$$

where access to f is limited to unbiased stochastic gradient estimates. The function f is assumed to be differentiable and L-smooth. While our theoretical analysis focuses on the strongly convex quadratic case, the proposed method is designed to operate in general smooth (and potentially nonconvex) settings.

This formulation is central in modern machine learning and large-scale optimization, particularly in training deep neural networks and large-scale empirical risk minimization. In these applications, full gradients are often unavailable or expensive to compute, making stochastic optimization essential. As a result, the design of stochastic optimization algorithms must carefully balance:

– **Efficiency**: achieving rapid convergence under noise,
– **Stability**: tolerating large learning rates and numerical inaccuracies,
– **Acceleration**: leveraging momentum or curvature information to improve convergence.

Although momentum-based methods such as Nesterov's acceleration are widely used, their behavior in the stochastic setting is not fully understood. In this work, we propose a new momentum method derived from a second-order stochastic differential equation (SDE) with damping, discretized using a semi-implicit Gauss-Seidel scheme. The resulting algorithm, called *NAG-GS*, is simple, stable, and empirically robust. Further section reviews related work on acceleration, ODE-inspired methods, and stochastic optimization.

1.1 Related work

Acceleration and ODE discretization. The perspective of interpreting optimization algorithms as discretizations of dynamical systems has become widely adopted in recent years [19,24,27,21]. This approach builds an explicit link between the properties of continuous-time ODEs and the behavior of discrete-time methods. A wide range of algorithms, including gradient descent, Nesterov acceleration, mirror descent [10], proximal algorithms [2], and ADMM [7], have been studied through this lens.

A key insight in this line of work is that the choice of discretization has a critical impact on algorithmic performance. For instance, Shi et al. [22] and Zhang et al. [31] analyzed optimal discretizations for various classes of ODEs, while Malladi et al. [18] extended this reasoning to stochastic first-order methods. Our method continues in this tradition but employs a novel semi-implicit Gauss-Seidel discretization that improves stability and robustness in stochastic optimization.

Laborde and Oberman [12] further extended Lyapunov techniques to the stochastic setting by interpreting stochastic Nesterov-type updates as perturbations of second-order ODEs. Their convergence analysis is based on discrete energy decay using fixed step sizes. In contrast, our method is derived from a new Gauss-Seidel discretization of a damped SDE, and we analyze its stability using spectral techniques and dynamical parameters. Moreover, our method requires only a single auxiliary vector and scales well to large-scale machine learning tasks.

Stochastic acceleration and AC-SA. The AC-SA method of Lan [13] achieves optimal convergence for structured stochastic convex problems via step-size scheduling and weighted averaging. However, AC-SA assumes a composite structure $f(x) + h(x)$ and uses multiple oracle calls per iteration. Our method, by contrast, handles general smooth objectives and is implementation-friendly, requiring only simple updates without averaging.

Luo and Chen [17] derive accelerated first-order methods from high-order ODE solvers and provide convergence guarantees for convex and strongly convex objectives. Their analysis is elegant but remains purely deterministic. In contrast, we extend the NAG-GS method to the stochastic setting, and develop a detailed convergence and stability analysis in the case of quadratic objectives using tools from numerical analysis, such as spectral radius and A-stability.

In the deterministic case, we provide a refined analysis for quadratic forms that yields optimal step sizes achieving the highest possible contraction rate. Moreover, we show that NAG-GS, for some specific choice of hyperparameters, reduces to the Quasi-Monotone Method (QMM) of [27] under constant damping, allowing us to derive an $\mathcal{O}(1/k)$ convergence rate for strongly convex but possibly non-smooth functions. This connection bridges our stochastic algorithm with optimal deterministic schemes and establishes a solid theoretical foundation for further extensions.

Empirical momentum methods. Momentum-based optimizers such as SGD with momentum, Nesterov's accelerated gradient, and Adam are widely used in deep learning due to their practical effectiveness. However, their theoretical behavior in the stochastic regime remains only partially understood, particularly in terms of stability under large learning rates and the role of momentum in nonconvex settings. While recent works such as [25,32] propose deterministic schemes with optimal convergence properties, they do not leverage continuous-time or SDE-based insights and are not directly applicable to stochastic optimization. Our method is grounded in a principled stochastic differential equation framework, demonstrating both strong empirical performance and robustness across a wide range of learning rates.

1.2 Contributions

We summarize our main contributions below:

- We propose a novel optimization algorithm, **NAG-GS**, derived from a semi-implicit Gauss-Seidel discretization of a second-order stochastic differential equation with damping.
- We analyze its convergence and numerical stability in the strongly convex quadratic case, derive optimal step sizes, and connect the method to the Quasi-Monotone Method [27] in the deterministic limit.
- We highlight the differences between NAG-GS and existing stochastic acceleration methods such as AC-SA [13], emphasizing the simplicity, generality, and empirical performance of our method.
- We validate the effectiveness of NAG-GS on both convex and nonconvex tasks, including training deep ResNet and Transformer architectures. The method supports significantly larger learning rates than existing optimizers while maintaining stability and accuracy.

1.3 Organization of the paper

The remainder of the paper is structured as follows: Section 1.4 introduces the theoretical preliminaries and notation used throughout the paper. In Section 2, we derive NAG-GS by discretizing a second-order stochastic differential equation (SDE) using a semi-implicit Gauss-Seidel scheme. The resulting method, called *NAG-GS*, is first analyzed in the quadratic case for both deterministic and stochastic settings. We also illustrate its behavior on a one-dimensional non-convex SDE, with strong empirical evidence of acceleration (included in the Supplementary Material due to the lack of theoretical guarantees in this regime). Section 3 presents a comprehensive experimental evaluation of NAG-GS, from low-dimensional logistic regression to large-scale machine learning tasks including ResNet-20, VGG-11, and Visual Transformer training. We now introduce the notation and theoretical preliminaries used throughout the analysis of NAG-GS.

1.4 Preliminaries

We start here with some general considerations in the deterministic setting for obtaining accelerated Ordinary Differential Equations (ODE) that will be extended in the stochastic setting in Section 2.1. We consider iterative methods for solving the unconstrained minimization problem:

$$\min_{x\in\mathbb{R}^n} f(x), \tag{2}$$

where $f:\mathbb{R}^n \to \mathbb{R}\cup\{+\infty\}$ is a proper closed convex extended real-valued function. In the following, for simplicity, we shall consider a function f smooth on the entire space. We also suppose $\mathbb{R}^n$ is equipped with the canonical inner product

$\langle x, y\rangle = \sum_{i=1}^{n} x_i y_i$ and the correspondingly induced norm $\|x\| = \sqrt{\langle x, x\rangle}$. Finally, we will consider in this section the class of functions $\mathcal{S}_{L,\mu}^{1,1}$ which stands for the set of strongly convex functions of parameter $\mu > 0$ with Lipschitz-continuous gradients with constant $L > 0$. For such class of functions, it is well-known that the global minimum exists and is unique [20]. One well-known approach to deriving the Gradient Descent (GD) method is discretizing the gradient flow:

$$\dot{x}(t) = -\nabla f(x(t)), \quad t > 0. \tag{3}$$

The simplest forward (explicit) Euler method with step size (also refer as learning rate) $\alpha_k > 0$ leads to the GD method

$$x_{k+1} \leftarrow x_k - \alpha_k \nabla f(x_k).$$

In numerical analysis, it is widely recognized that this method is conditionally A-stable. It means that the convergence appears only for $\alpha_k \in \mathcal{I}$, where $\mathcal{I}$ denotes the interval of the positive half line. More formal definition of the conditional A-stability is presented in Section 1.5.1. Moreover, when considering $f \in \mathcal{S}_{L,\mu}^{1,1}$ with $0 \leq \mu \leq L \leq \infty$, the utilization of a step size $\alpha_k = 1/L$ leads to a linear convergence rate. It is important to highlight that the highest rate of convergence is attained when $\alpha_k = \frac{2}{\mu+L}$. In such a scenario, we have

$$\|x_k - x^\star\|^2 \leq \left(\frac{Q_f - 1}{Q_f + 1}\right)^{2k} \|x_0 - x^\star\|^2, \tag{4}$$

where Q_f is defined as $Q_f = \frac{L}{\mu}$ and is commonly referred to as the condition number of a function f [20]. Another approach that can be considered is the backward (implicit) Euler method, which is represented as:

$$x_{k+1} \leftarrow x_k - \alpha_k \nabla f(x_{k+1}), \tag{5}$$

where learning rate $\alpha_k > 0$. This method is unconditionally A-stable. In a nutshell, A-stability in numerical ordinary differential equations characterizes a method's performance in the asymptotic regime as time approaches infinity. An unconditionally A-stable method is one where the step size can be arbitrarily large, yet the global error of the method converges to zero. We give more details about the notion in Section 1.5. Hereunder, we summarize the methodology proposed by [17] to come up with a general family of accelerated gradient flows by focusing on the following simple problem:

$$\min_{x \in \mathbb{R}^n} f(x) = \frac{1}{2} x^\top A x \tag{6}$$

for which the gradient flow in (3) reads as:

$$\dot{x}(t) = -Ax(t), \quad t > 0, \tag{7}$$

where $A \in \mathbb{R}^{n \times n}$ is symmetric positive semi-definite matrix ensuring that $f \in \mathcal{S}_{L,\mu}^{1,1}$ where μ and L respectively correspond to the minimum and maximum eigenvalues of matrix A, which are real and positive by assumption. Instead of directly resolving (7), authors of [17] opted to address a general linear ODE system as follows:

$$\dot{y}(t) = Gy(t), \quad t > 0. \tag{8}$$

The central concept is to search for a system (8) with an asymmetric block matrix G that transforms the spectrum of A from the real line to the complex plane, reducing the condition number from $\kappa(A) = \frac{L}{\mu}$ to $\kappa(G) = O\left(\sqrt{\frac{L}{\mu}}\right)$. Subsequently, accelerated gradient methods can be constructed from A-stable methods to solve (8) with a significantly larger step size, improving the contraction rate from $O\left(\left(\frac{Q_f-1}{Q_f+1}\right)^{2k}\right)$ to $O\left(\left(\frac{\sqrt{Q_f}-1}{\sqrt{Q_f}+1}\right)^{2k}\right)$. Moreover, to handle the convex case $\mu = 0$, the authors in [17] combine the transformation idea with a suitable time scaling technique. In this paper, we consider one transformation that relies on the embedding of A into some 2×2 block matrix G with a rotation built-in [17]:

$$G_{NAG} = \begin{bmatrix} -I & I \\ I \cdot \mu/\gamma - A/\gamma & -I \cdot \mu/\gamma \end{bmatrix}, \tag{9}$$

where γ is a positive time scaling factor that satisfies

$$\dot{\gamma}(t) = \mu - \gamma(t), \quad \gamma(0) = \gamma_0 > 0. \tag{10}$$

Denote by $\Re(\lambda)$ and $\sigma(G)$ the real part of λ and spectrum of matrix G, respectively. Then, given A positive definite, we can easily show that for the considered transformation, we have that $\Re(\lambda) < 0$ for all $\lambda \in \sigma(G)$. Further, we will denote by $\rho(G) := \max_{\lambda \in \sigma(G)} |\lambda|$ the spectral radius of matrix G. Let us now consider the NAG block matrix G_{NAG} and let $y = (x, v)$, the dynamical system given in (8) with $y(0) = y_0 \in \mathbb{R}^{2n}$ reads:

$$\begin{aligned} \frac{dx}{dt} &= v - x, \\ \frac{dv}{dt} &= \frac{\mu}{\gamma}(x - v) - \frac{1}{\gamma}Ax \end{aligned} \tag{11}$$

with initial conditions $x(0) = x_0$ and $v(0) = v_0$. Note that this linear ODE can be expressed as:

$$\gamma\ddot{x} + (\gamma + \mu)\dot{x} + Ax = 0, \tag{12}$$

where Ax is therefore the gradient of f w.r.t. x. Thus, one could generalize this approach for any function $f \in \mathcal{S}^{1,1}_{L,\mu}$ by replacing Ax by $\nabla f(x)$, respectively, within (9), (11) and (12). Since the notion of A-stability is crucial for the derivation of the main result, we provide a brief intro to this notion in the next section and highlight the dependence of the feasible step size interval on the A-stability of the method.

1.5 Insights about the notion of A-stability

Before we recall the notion of A-stability of ODE solvers, note that this paper only considers ODEs with asymptotically stable solutions. In particular, let the Jacobian matrix of a function f at $x(t)$ be $J_f(x(t))$, then we focus on ODE

$$\dot{x}(t) = f(t, x(t)), \quad x(0) = x_0, \tag{13}$$

with $\Re(\lambda) < 0$ for $\lambda \in \sigma(J_f(x(t)))$, where $\Re$ denotes real part of a complex number.

Illustrative toy example. Consider an objective function $\Phi(x) := \frac{1}{2}x^T Ax$ with A being symmetric positive definite such that:

$$0 < \mu := \lambda_{\text{min}}(A) \leq \lambda \leq \lambda_{\text{max}}(A) =: L \leq +\infty, \quad \forall \lambda \in \sigma(A).$$

In this example, the global minimum is achieved at $x^\star = 0$ and $\nabla \Phi(x) = Ax$.

Equation (13) boils down to the linear ODE:

$$\dot{x}(t) = -Ax(t), \quad x(0) = x_0. \tag{14}$$

Here $J_f = -A$, hence $\Re(\lambda) < 0$ for all $\lambda \in \sigma(J_f)$ and it is not hard to show that $\|x(t)\|_2 \to 0$ as $t \to \infty$. Note that $x^\star = 0$ is an equilibrium solution of the dynamical system (14).

1.5.1 A-Stability of ODE solver

Now we briefly recall the concept of A-stability for ODE solvers, which originates from the classical notion of stability based on eigenvalues having negative real parts, as introduced by Dahlquist [5].

To avoid the general and potentially lengthy discussion of A-stability for nonlinear ODEs of the form

$$\dot{x}(t) = f(t, x(t)), \quad x(0) = x_0, \quad \text{with } \Re(\lambda) < 0 \text{ for all } \lambda \in \sigma(J_f),$$

we consider instead the simplified case of a linear autonomous system:

$$\dot{x}(t) = Gx(t), \quad x(0) = x_0, \quad \text{with } \Re(\lambda) < 0 \text{ for all } \lambda \in \sigma(G). \tag{15}$$

A one-step numerical method ϕ for solving equation 15 with step size $\alpha > 0$ generates iterates of the form:

$$x_{k+1} = E_\phi(G, \alpha)\, x_k,$$

where $E_\phi(G, \alpha) \in \mathbb{R}^{n \times n}$ is called the *stability matrix* (also known as the *iteration matrix*) associated with the method ϕ, the step size α, and the system matrix G. This matrix governs the spectral behavior of the numerical method and determines whether the iterates converge to zero.

The numerical scheme ϕ is called absolute stable or A-stable if $\rho(E_\phi(G, \alpha)) < 1$ (from which the asymptotic convergence $x_k \to 0$ follows). If $\rho(E_\phi(G, \alpha)) < 1$ holds for all $\alpha > 0$, then the scheme is called unconditionally A-stable, and if $\rho(E_\phi(G, \alpha)) < 1$ holds for some $\alpha \in \mathcal{I}$, where $\mathcal{I}$ denotes an interval of the positive half line, then the scheme is conditionally A-stable. The following subsection considers two popular ODE solvers from the A-stability perspective.

1.5.2 Explicit and implicit Euler schemes

Here, we review the stability of the explicit and implicit Euler schemes for solving (15). The analytical solution for (15) with a constant discretization step α generates the iterates:

$$x_{k+1} = x_k + \int_{t_k}^{t_{k+1}} Gx(s)ds, \quad k = 0, \ldots, M-1.$$

For $G = -A$, the explicit Euler method approximates the integral by the area of a rectangle with width α and height $-Ax_k$. This leads to the iterates $x_{k+1} = (I - \alpha A)x_k$ which corresponds to the GD scheme for minimizing $\Phi(x) = \frac{1}{2}x^T Ax$. The explicit Euler method is A-stable if the spectral radius of $I - \alpha A$ is strictly less than 1, i.e., if $\rho(I - \alpha A) = \max_{\lambda \in \sigma(I - \alpha A)} |\lambda| < 1$. We can easily show that $\rho(I - \alpha A) =$ max $(|1 - \alpha\mu|, |1 - \alpha L|)$, where μ and L respectively denote the smallest and largest eigenvalue of A. Therefore, the explicit Euler method is A-stable if $0 < \alpha < 2/L$. Additionally, we can determine the optimal α that minimizes the spectral radius: $\min_{\alpha > 0} \rho(I - \alpha A)$, which gives $\alpha^\star = 2/(\mu + L)$, resulting in $\rho(I - \alpha^\star A) = (Q_f - 1)/(Q_f + 1)$. Assuming $0 < \mu \leq L < \infty$, we have $0 < \alpha^\star = 2/(\mu + L) < 2/L$. Hence, the explicit Euler method is *conditionally* A-stable, and the norm convergence with a linear rate follows as in (4).

On the other hand, the implicit Euler scheme approximates the integral by the area of a rectangle with a height of $-Ax_{k+1}$, leading to the iterates $x_{k+1} = (I + \alpha A)^{-1}x_k$. The term $\rho(I + \alpha A)^{-1}$ can be expressed as max $\left(|\frac{1}{1+\alpha\mu}|, |\frac{1}{1+\alpha L}|\right)$. This implies that the stability condition $\rho(I + \alpha A)^{-1} < 1$ holds true for all $\alpha > 0$, making the implicit Euler scheme *unconditionally* A-stable. Moreover, the implicit Euler method can achieve a faster convergence rate by time rescaling, as it is not limited by any constraints on the step size. This is equivalent to opting for a larger step size.

At the same time, even for the quadratic objective function $\Phi(x) = \frac{1}{2}x^\top Ax$, it is clear that the implicit Euler scheme is more computationally intensive since it requires solving a sequence of linear systems. Therefore, straightforward usage of this scheme in large-scale setups is challenging. To avoid this issue, we focus on semi-implicit schemes that combine the benefits of explicit and implicit Euler schemes. In particular, we consider Gauss-Seidel semi-implicit discretization of the accelerated system (11) to derive the novel NAG-GS optimization method. Extension of the NAG-GS to the inexact gradient setup is obtained through perturbation of the right-hand side in (11) with a continuous Ito martingale. This perturbation converts the deterministic system of ODEs to the system of accelerated Stochastic Differential Equations (SDEs), see Section 2.2.

2 Model and Theory

2.1 Accelerated Stochastic Gradient flow

In section 1.4, we have presented a family of accelerated Gradient flows obtained by an appropriate spectral transformation G of the matrix A, see (11). When generalizing from the quadratic to the general convex case, the term Ax in (11) is replaced by $\nabla f(x)$.

Within the context of training neural networks, function $f(x)$ corresponds to some loss function, and its gradient $\nabla f(x)$ is contaminated by noise due to a finite-sample gradient estimate. The study of accelerated Gradient flows is now adapted to include and model the effect of the noise; to achieve this, we consider the dynamics given in (8) perturbed by a general martingale process. This leads us to consider the following Accelerated Stochastic Gradient (ASG) flow:

$$\begin{aligned} \frac{dx}{dt} &= v - x, \\ \frac{dv}{dt} &= \frac{\mu}{\gamma}(x - v) - \frac{1}{\gamma}Ax + \frac{dZ}{dt}, \end{aligned} \tag{16}$$

which corresponds to an (Accelerated) system of SDEs, where $Z(t)$ is a continuous Itô martingale. We assume that $Z(t)$ has the simple expression $dZ = \sigma dW$, where $W = (W_1, \dots, W_n)$ is a standard n-dimensional Brownian Motion. In this paper, we consider only the case of constant parameter σ, this parameter is sometimes referred to as the diffusion coefficient in SDE. For the sake of completeness, let us recall the definition of a Brownian motion:

Definition 2.1 (Brownian Motion) A standard n-dimensional Brownian motion $W = (W_t)_{t\geq 0}$ is a stochastic process with the following properties:

- $W_0 = 0$ almost surely,
- The paths $t \mapsto W_t$ are almost surely continuous,
- The process has **independent increments**: for any $0 \leq t_0 < t_1 < \cdots < t_k$, the random variables $W_{t_1} - W_{t_0}, \dots, W_{t_k} - W_{t_{k-1}}$ are independent,
- The process has **Gaussian increments**: for any $s < t$, the increment $W_t - W_s$ is normally distributed with mean zero and covariance $(t - s)I_n$, i.e., $W_t - W_s \sim \mathcal{N}(0, (t - s)I_n)$.

Brownian motion is also referred to as a Wiener process, and it is the canonical model of continuous stochastic noise used in stochastic differential equations.

The next section presents the discretizations corresponding to the ASG flow given in (16).

2.2 Discretization: Gauss-Seidel Splitting and Semi-Implicitness

This section presents the primary strategy to discretize the Accelerated SDE's system (16). The motivation behind the discretization method is to derive integration schemes that are, in the best case, unconditionally A-stable or conditionally A-stable with the highest possible integration step. In the classical terminology of (discrete) optimization methods, this value ensures convergence of the obtained methods with the largest possible step size. Consequently, it improves the contraction rate (or the rate of convergence). In Section 1.4, we have briefly recalled that the most well-known unconditionally A-stable scheme is the backward Euler method (5), which is an implicit method and hence can achieve a faster convergence rate. However, this requires solving a linear system or, in the case of a general convex function, computing the root of a non-linear equation, leading to a high computational cost. This is why few implicit schemes are used to solve high-dimensional optimization problems. Still, an explicit scheme closer to the implicit Euler method is expected to have good stability with a larger step size than a forward Euler method. Furthermore, assuming a Gaussian noise process, it is crucial to propose a solver capable of handling a wide range of step size values. Specifically, allowing for a larger ratio α/b, where b denotes the mini-batch size, increases the likelihood of converging to wider local minima, ultimately enhancing the generalization performance of the trained model [9]. More discussion of this aspect is presented in Supplementary materials.

Motivated by the Gauss-Seidel (GS) method for solving linear systems, we consider the matrix splitting $G = M + N$ with M being the lower triangular part of G and $N = G - M$, we propose the following Gauss-Seidel splitting scheme for (8) perturbed with noise:

$$\frac{y_{k+1} - y_k}{\alpha_k} = My_{k+1} + Ny_k + \begin{bmatrix} 0 \\ \sigma \frac{W_{k+1}-W_k}{\alpha_k} \end{bmatrix} \tag{17}$$

which for $G = G_{NAG}$ (see (9)), gives the following semi-implicit scheme with step size $\alpha_k > 0$:

$$\begin{aligned} \frac{x_{k+1} - x_k}{\alpha_k} &= v_k - x_{k+1}, \\ \frac{v_{k+1} - v_k}{\alpha_k} &= \frac{\mu}{\gamma_k}(x_{k+1} - v_{k+1}) - \frac{1}{\gamma_k} Ax_{k+1} + \sigma \frac{W_{k+1} - W_k}{\alpha_k}. \end{aligned} \tag{18}$$

Note that due to the properties of Brownian motion, we can simulate its values at the selected points by: $W_{k+1} = W_k + \Delta W_k$, where ΔW_k are independent random variables with distribution $\mathcal{N}(0, \alpha_k)$.

Furthermore, ODE (10) corresponding to the parameter γ is also discretized implicitly:

$$\frac{\gamma_{k+1} - \gamma_k}{\alpha_k} = \mu - \gamma_{k+1}, \quad \gamma_0 > 0. \tag{19}$$

As already mentioned earlier, heuristically, for general $f \in \mathcal{S}^{1,1}_{L,\mu}$ with $\mu \geq 0$, we just replace Ax in (18) with $\nabla f(x)$ and obtain the following NAG-GS scheme:

$$\begin{aligned} \frac{x_{k+1} - x_k}{\alpha_k} &= v_k - x_{k+1}, \\ \frac{v_{k+1} - v_k}{\alpha_k} &= \frac{\mu}{\gamma_k}(x_{k+1} - v_{k+1}) - \frac{1}{\gamma_k} \nabla f(x_{k+1}) + \sigma \frac{W_{k+1} - W_k}{\alpha_k}. \end{aligned} \tag{20}$$

Finally, we introduce the NAG-GS method (see Algorithm 1). In this method, we consider the presence of unknown noise when computing the gradient $\nabla f(x_{k+1})$. We denote this noisy gradient as $\nabla \tilde{f}(x_{k+1})$ in Algorithm 1. Notably, to achieve strict equivalence with the scheme described in (20), we have the relationship $\nabla \tilde{f}(x_{k+1}) = \nabla f(x_{k+1}) + \sigma\mu(1 - \frac{1}{b_k})(W_{k+1} - W_k)$, where b_k is defined as $b_k := \alpha_k\mu(\alpha_k\mu + \gamma_k)^{-1}$.

Algorithm 1 Nesterov Accelerated Gradients with Gauss–Seidel splitting (NAG-GS).

```
Input: Choose point $x_0 \in \mathbb{R}^n$, some $\mu \geq 0, \gamma_0 > 0$.
  Set $v_0 := x_0$.
  for $k = 1, 2, \ldots$ do
    Choose step size $\alpha_k > 0$.
    ▷ Update parameters and state $x$:
    Set $a_k := \alpha_k(\alpha_k + 1)^{-1}$.
    Set $\gamma_{k+1} := (1 - a_k)\gamma_k + a_k \mu$.
    Set $x_{k+1} := (1 - a_k)x_k + a_k v_k$.
    ▷ Update state $v$:
    Set $b_k := \alpha_k \mu(\alpha_k \mu + \gamma_k)^{-1}$.
    Set $v_{k+1} := (1 - b_k)v_k + b_k x_{k+1} - \mu^{-1} b_k \nabla \tilde{f}(x_{k+1})$.
  end for
```

Remark 2.1 (Complexity of NAG-GS vs. AdamW) According to Algorithm 1, the NAG-GS algorithm requires *one* auxiliary vector with a dimension equal to the number of the trained parameters. In contrast, AdamW requires *two* auxiliary vectors of the same dimension and updates them respectively. Hence, NAG-GS has lower computational complexity and memory requirements than AdamW.

Moreover, the step size update can be performed using different strategies. For instance, one may choose the method proposed in [20, Method 2.2.7] which specifies to compute $\alpha_k \in (0, 1)$ such that $L\alpha_k^2 = (1 - \alpha_k)\gamma_k + \alpha_k \mu$. Note that for $\gamma_0 = \mu$, hence the sequences $\gamma_k = \mu$ and $\alpha_k = \sqrt{\frac{\mu}{L}}$ for all $k \geq 0$. In Section 2.3, we discuss how to compute the step size for Algorithm 1.

We have also considered full-implicit discretizations. However, interest in such methods is limited to ML applications since the obtained implicit schemes use second-order information about f, and such schemes are typically intractable for real-life ML models. Therefore, we analyze full-implicit discretizations in the Supplementary materials (Section 3).

2.3 Convergence analysis of quadratic case

We propose to study how to select a maximum step size that ensures an optimal contraction rate while guaranteeing the convergence or the stability of the NAG-GS method once used to solve SDE's system (16). Ultimately, we show that the choice of the optimal step size is mainly influenced by the values of μ, L, and γ. These hyperparameters are central, and to show this, we study two key quantities, namely the spectral radius of the iteration matrix and the covariance matrix associated with the NAG-GS method summarized by Algorithm 1. Note that this theoretical study only concerns the case $f(x) = \frac{1}{2}x^\top Ax$. For the reader's convenience, we provide the main theoretical result below and place the proof in Supplementary materials.

Theorem 2.1 *For G_{NAG} (9), given $\gamma \geq \mu$, and assuming $0 < \mu = \lambda_1 \leq \ldots \leq \lambda_n = L < \infty$; if $0 < \alpha \leq \frac{\mu+\gamma+\sqrt{(\mu-\gamma)^2+4\gamma L}}{L-\mu}$, then the NAG-GS method summarized by Algorithm 1 is convergent for the n-dimensional case, with $n > 2$.*

The plan of the proof is the following. First, we provide the full analysis of the spectral radius of the iteration matrix associated with the NAG-GS method and the covariance matrix at stationarity w.r.t. hyperparameters μ, L, γ and σ for the dimension $n = 2$. Second, based on such analysis, we present the step size that optimizes the contraction rate. Finally, we extend the proof for $n = 2$ to larger dimensions and obtain Theorem 2.1. In addition, we perform illustrative numerical simulations to confirm the obtained theoretical result for the quadratic objective function.

Remark 2.2 It is essential to mention that the optimal contraction rate of NAG-GS aiming at minimizing a strongly convex quadratic function is reached for $\alpha^* = \frac{\gamma+\mu+\sqrt{(\gamma-\mu)^2+4\gamma L}}{L-\mu}$.

Remark 2.3 (Convergence analysis in the deterministic setting) When NAG-GS is applied using the exact gradient of f (as opposed to a stochastic estimate), we refer to this as the deterministic setting. To our knowledge, no formal convergence rate was previously established for NAG-GS when applied to general strongly convex or convex functions. While the convergence theory in the convex (non-strongly convex) case remains open, we provide here a partial resolution in the strongly convex setting.

In [28], the authors introduce a unifying Lyapunov framework for analyzing the non-asymptotic convergence of momentum-based optimization algorithms. They show that the classical technique of estimating sequences is equivalent to a particular class of Lyapunov functions, both in continuous and discrete time. This connection allows them to systematically derive convergence guarantees for various discretization schemes of second-order differential equations. In particular, they propose a family of algorithms called *Quasi-Monotone Methods* (QMM), characterized by the following update rules for each iteration $k \geq 0$ with a unitary time step δ:

$$\boxed{\begin{aligned} x_{k+1} &= \frac{\tau_k}{1+\tau_k} v_k + \frac{1}{1+\tau_k} x_k, \\ \nabla h(v_{k+1}) - \nabla h(v_k) &= \tau_k \left(\nabla h(x_{k+1}) - \nabla h(z_{k+1}) - \frac{1}{\mu} \nabla f(x_{k+1}) \right) \end{aligned}} \tag{QMM Iterates}$$

Now, in the Euclidean case $h(x) = \frac{1}{2}\|x\|^2$, we have $\nabla h = \mathrm{id}$. Identifying $\alpha_k = \tau_k$ and fixing the NAG-GS parameter $\gamma_0 = \mu$ so that $\gamma_k = \mu$ for all $k \geq 0$, the QMM and NAG-GS iterates become algebraically equivalent. To the best of our knowledge, this equivalence is not explicitly noted in the Lyapunov or ODE-based literature such as [17].

Furthermore, [28] propose the discrete-time Lyapunov function

$$E_k = A_k \left(f(x_k) - f(x^\star) + \mu D_h(x^\star, v_k) \right),$$

and under the assumption that f is μ-strongly convex with respect to h, and that h is η-strongly convex, they prove the bound:

$$E_{k+1} - E_k \leq \frac{A_k \tau_k^2}{2\mu\eta} \|\nabla f(x_{k+1})\|^2.$$

By choosing the discrete-time sequence $A_k = \frac{(k+2)(k+3)}{6}$ and step parameter $\tau_k = \frac{2}{k+2}$, which satisfies the consistency condition $\frac{A_{k+1}-A_k}{A_k} = \tau_k$, and assuming bounded gradients (for instance by assuming that the function f is Lipschitz continuous), this leads to the convergence rate

$$f(x_k) - f(x^\star) \le \mathcal{O}\left(\frac{1}{k}\right).$$

Although this rate is not optimal for smooth, strongly convex functions (where accelerated methods achieve linear convergence), it is optimal for nonsmooth, strongly convex optimization using subgradient methods. Notably, the result requires neither smoothness nor Lipschitz continuity, further illustrating the robustness of QMM and, by extension, the NAG-GS method. This connection suggests that NAG-GS can serve as a robust optimizer even for general (possibly nonsmooth) strongly convex functions, and opens the door to extending Lyapunov-based techniques to convex and stochastic settings.

3 Experiments

We test the NAG-GS method on several neural architectures: logistic regression, transformer model for natural language processing (RoBERTa model) and computer vision (ViT model) tasks, and residual networks for computer vision tasks (ResNet20). To ensure a fair benchmark of our method on these neural architectures, we replace only the optimizer with NAG-GS while preserving all other hyperparameters of the training process. Our experiments can be reproduced using the source code[1].

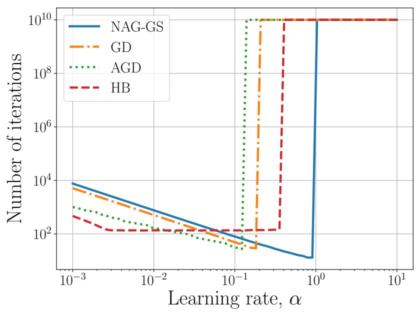
3.1 Toy problem

We compare the convergence of the NAG-GS method for a strongly convex quadratic function with other first-order methods including standard accelerated methods like Heave Ball (HB) method and Accelerated Gradient Descent (AGD). We select these methods since they establish an optimal convergence rate for the considered type of problems. This test demonstrates that NAG-GS can work with larger learning rates. If a large learning rate is used, NAG-GS converges faster than competitors with smaller learning rates.

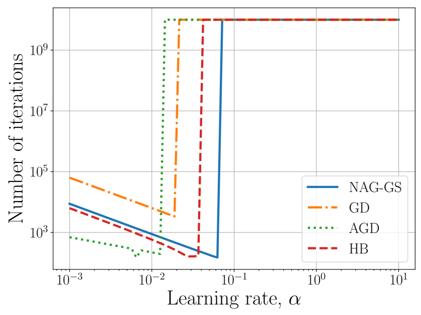
Strongly convex quadratic function. Consider the problem $\min_x f(x)$, where $f(x) = \frac{1}{2}x^\top Ax - b^\top x$ is convex quadratic function. The matrix $A \in \mathbb{S}^n_{++}$ is symmetric and positive semidefinite, $L = \lambda_{\max}(A)$, $\mu = \lambda_{\min}(A)$ and $n = 100$. We assume that method converges if $f(x_k) - f^* \le 10^{-4}$, where $f^* = f(x^*)$ is the optimum function value. If some learning rate leads to divergence, we set the number of iterations to 10^{10}. In this experiment, we use the accelerated gradient descent (AGD) version from [24].

[1] `https://github.com/naggsopt/naggs`

In NAG-GS we use constant $\gamma = \mu = \lambda_{\min}(A)$. In the HB method, we use $\beta = 0.9$. Also, we test 70 learning rates distributed uniformly in the logarithmic grid in the interval $[10^{-3}, 10]$. Figure 1 shows the dependence of the number of iterations needed for convergence of NAG-GS, gradient descent (GD), and accelerated gradient descent (AGD) on the learning rates for different μ and L. We observe that NAG-GS provides two benefits. First, it converges for larger learning rates than the GD, AGD, and HB methods. Second, in the large learning rate regime, NAG-GS converges faster in terms of the number of iterations than GD and AGD. Although the AGD method is optimal among the first-order methods, its feasible learning rate is strictly bounded by $1/L$. In contrast, the non-asymptotic convergence of the NAG-GS method remains a subject of future work. However, NAG-GS with a large learning rate, which is infeasible for the AGD method, shows faster convergence even for the ill-conditioned problem, as shown in Figure 1b.



(a) $\mu = 1$, $L = 10$



(b) $\mu = 10^{-1}$, $L = 100$

Fig. 1: Dependence of the number of iterations needed for convergence on the learning rate. NAG-GS is more robust with respect to the learning rate than gradient descent (GD) and accelerated gradient descent (AGD). Also, NAG-GS converges faster than competitors if the learning rate is sufficiently large. The number of iterations 10^{10} indicates the divergence.

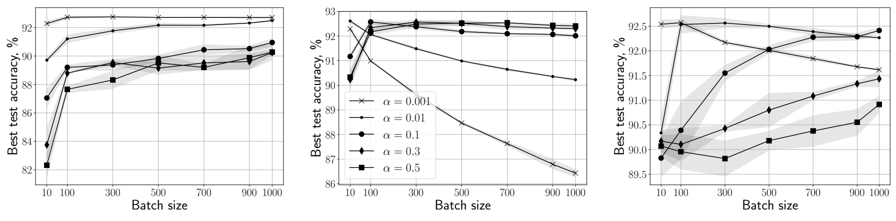
3.2 Logistic regression

In this section, we benchmark the NAG-GS method against SGD with momentum (SGD-M) and AdamW optimizers on the multiclass logistic regression model and MNIST dataset [14]. We select such competitors since they are the standard methods of choice in training deep neural networks. SGD-M typically shows better generalization, while AdamW automatically adjusts the learning rate and provides more stable convergence. Since this problem is convex and non-quadratic, we consider this problem as the natural next test case after the theoretical analysis of the NAG-GS method in Section 2.3 and numerical tests for the quadratic convex problem. We use the following hyperparameters: batch size is 500, momentum term in SGD-M equals 0.9, $\mu = 1$ and $\gamma = 1$ in NAG-GS, and no weight decay in SGD-M and AdamW, i.e., AdamW coincides with Adam optimizer. In Table 1, we present the learning rates, final test accuracy, and the number of epochs necessary for convergence for every optimizer. We assume that the training process has converged if the change in the test accuracy is minor. We observe that the larger the learning rate, the better the performance of the NAG-GS. Note that the highest test accuracy is achieved by competitors only after 20 or 40 epochs, while NAG-GS with a larger learning rate achieves a similar performance already after ten epochs. At the same time, we observe that the constant small learning rate leads to poor performance of the NAG-GS method. These observations motivate the use of the NAG-GS optimizer in the first stage of training to achieve high test accuracy after a small number of epochs. If this test accuracy is still insufficient, switching to another optimizer or decreasing the learning rate could be a good strategy for improving performance. In this section, we confirm numerically that the NAG-GS method allows larger learning rate values than the SGD-M and AdamW optimizers. Moreover, the results indicate that the semi-implicit nature of the NAG-GS method indeed ensures the acceleration effect through the use of larger learning rates while preserving the high performance of the model. This behavior holds not only for convex quadratic problems but also for non-quadratic convex ones.

Table 1: Test accuracies for NAG-GS, SGD-M, and AdamW that train the multiclass logistic regression model. The reported accuracies are averaged over three random seeds, and standard deviations are also presented. NAG-GS with a learning rate $\alpha = 0.5$ converges to the highest test accuracy after only 10 epochs, while the performance of AdamW is similar only after 40 epochs and uses $\alpha = 10^{-3}$. We highlight the best test accuracy in bold and underline the second-best result.

Learning rate, α	# epochs	Optimizer	Best test accuracy
10^{-3}	40	NAG-GS	88.47 ± 0.05
		SGD-M	$\underline{92.01 \pm 0.04}$
		AdamW	$\mathbf{92.71 \pm 0.01}$
10^{-2}	20	NAG-GS	90.99 ± 0.02
		SGD-M	$\mathbf{92.50 \pm 0.04}$
		AdamW	$\underline{92.17 \pm 0.15}$
0.1	15	NAG-GS	$\mathbf{92.18 \pm 0.07}$
		SGD-M	$\underline{92.03 \pm 0.08}$
		AdamW	89.83 ± 0.24
0.5	10	NAG-GS	$\mathbf{92.38 \pm 0.02}$
		SGD-M	$\underline{90.04 \pm 0.35}$
		AdamW	88.66 ± 0.99

To solve the multiclass classification problem using the logistic regression model, we employ stochastic gradient estimation in all considered optimizers. Therefore, the effect of the batch size used in gradient estimation is important for understanding the robustness of the optimizers to the noise in the gradient estimation. To illustrate the effect of batch size on test accuracy, we simultaneously vary the batch size and the magnitude of the learning rate for the SGD-M, AdamW, and NAG-GS optimizers. The number of epochs for a particular learning rate is the same as presented in Table 1. Figure 2 demonstrates how best test accuracy depends on the batch size for a range of learning rates. It highlights that the NAG-GS optimizer achieves a higher best test accuracy than SGDM and AdamW in cases where a large learning rate and a large batch size are used. This feature makes NAG-GS preferable for utilizing GPU resources effectively in contrast to SGD-M, whose performance drops with larger batch sizes. Additionally, we observe that AdamW's performance degrades uniformly with large learning rates across all considered batch sizes. Thus, we show that NAG-GS achieves superior test accuracy in the multiclass classification problem in cases where a large learning rate and a large batch size are used. This regime is relevant to the considered problem since a large batch size improves GPU utilization and a large learning rate yields faster convergence, as shown in Table 1.



(a) AdamW (b) NAG-GS (c) SGDM

Fig. 2: Dependence of the best test accuracy on the batch size for various learning rates. The legend for plot b) works for plots a) and c). NAG-GS demonstrates robustness to large learning rates and batch sizes in contrast to the competitors.

3.3 VGG-11 and ResNet-20

We compare NAG-GS and SGD with momentum and weight decay (SGD-MW) on VGG-11 [23] and ResNet-20 [8] models. We do not consider AdamW here since it consumes more memory than SGD-MW, and this drawback becomes significant for these models. Training these models leads to non-convex stochastic optimization problems, which appear to be the next complexity level for the testing NAG-GS optimizer. Below, we demonstrate numerically the superior performance of NAG-GS in the first epochs of training the considered models using a large learning rate.

VGG-11. We test the VGG-11 model on the CIFAR-10 image classification problem [11] and demonstrate the robustness of NAG-GS to large learning rates compared to SGD-MW. The hyperparameters are the following: batch size equals 100, and the number of epochs is 50. We use the constant $\gamma = 1.$ and $\mu = 10^{-4}$ for NAG-GS, momentum term equal 0.9 and weight decay is 10^{-4} in SGD-MW. In comparison, we use two approaches: the best test accuracy achieved after 50 epochs and the convergence in the first epochs.

Table 2: Best test accuracies averaged over three random seeds for NAG-GS and SGD-MW optimizers. We use the VGG11 model on the CIFAR10 dataset for 50 epochs. NAG-GS achieves higher test accuracy for larger learning rates (in bold), while SGD-MW diverges for them. SGD-MW achieves slightly smaller test accuracy for the learning rate 0.01. Dash means divergence.

Learning rate, α	Optimizer	Best test accuracy
10^{-3}	NAG-GS	66.15 ± 0.81
	SGD-MW	78.84 ± 0.21
10^{-2}	NAG-GS	78.60 ± 0.40
	SGD-MW	81.58 ± 0.23
0.1	NAG-GS	81.50 ± 0.14
	SGD-MW	–
0.15	NAG-GS	$\mathbf{81.75 \pm 0.06}$
	SGD-MW	–

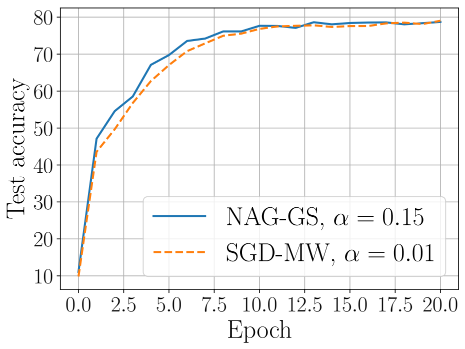


Fig. 3: Comparison of convergence NAG-GS and SGD-MW with different learning rates. NAG-GS achieves higher test accuracy faster than SGD-MW (see 1–10 epochs) while converging to a similar test accuracy during the middle of training. These lines are averaged over three random seeds, and the standard deviation is negligible and not plotted.

The first approach is illustrated in Table 2, where we show the best test accuracy achieved during training. From this table, it follows that NAG-GS achieved higher best test accuracy than SGD-MW due to the larger learning rate. The second approach is presented in Figure 3, where one can see that NAG-GS with a large learning rate ($\alpha = 0.15$) converges faster in the first ten epochs than SGD-MW with a smaller learning rate. We show only the first 20 epochs instead of the complete 50 epochs to highlight the gap at the beginning of the training. In further epochs, the test accuracy given by NAG-GS and SGD-MW remains similar.

Similar to the logistic regression model, we include the analysis of batch size effect on the best test accuracy for the NAG-GS and SGD-MW optimizers. Figure 4 demonstrates that NAG-GS converges for large learning rates in contrast to the SGD-MW optimizer. We also observe that a large learning rate regime for NAG-GS preserves the best test accuracy as the batch size increases. Thus, NAG-GS behavior in training a non-convex VGG11 model looks similar to the training of the convex logistic regression model.

ResNet-20. We carried out intensive experiments to evaluate the performance of NAG-GS in training residual networks. In particular, we use the ResNet-20 model and the CIFAR-10 dataset and the corresponding parameters reported in the literature. The experimental setup is the same except for the optimizer and its parameters. The best test score for NAG-GS is achieved for $\alpha = 0.11$, $\gamma = 17$, and $\mu = 0.01$. The search space for these hyperparameters are $\alpha \sim \mathrm{LogUniform}(10^{-2}, 10^{2})$, $\gamma \sim$

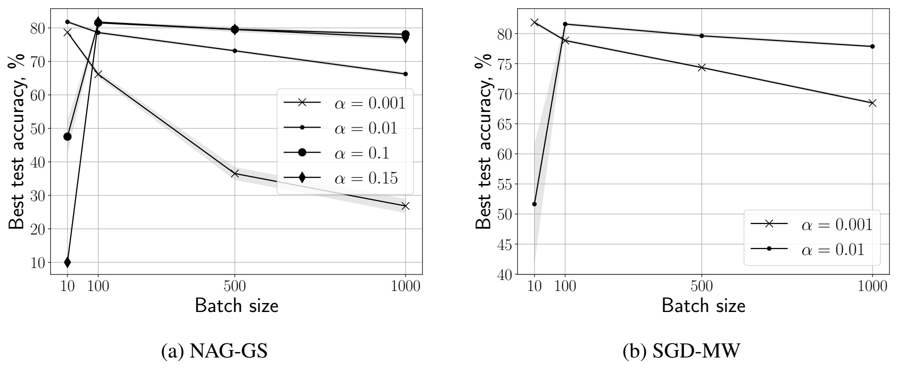


Fig. 4: Dependence of the best test accuracy on the batch size for various learning rates, VGG11 model and CIFAR-10 dataset. The number of epochs is 50. NAG-GS converges over a larger range of learning rates, and its performance drops less for larger learning rates as the batch size increases.

$\mathrm{LogUniform}(10^{-2}, 10^{2})$ and $\mu \sim \mathrm{Uniform}(-10, 100)$, where $\sim$ denotes the sampling procedure used in the Optuna library [1] to search the optimal hyperparameters. In addition, $\mathrm{LogUniform}(a, b)$ and $\mathrm{Uniform}(a, b)$ denote the log-uniform and uniform distributions on the given segments, respectively. We compare the convergence of NAG-GS and SGD-MW in terms of loss and test accuracy in Figure 5. The proposed NAG-GS optimizer provides better test accuracy than SGD-MW during the first 150 epochs. In the following epochs, the performance of NAG-GS and SGD-MW becomes similar. This observation confirms that the potential application of the NAG-GS optimizer is to improve performance in the first epochs, thereby speeding up convergence and reducing costs for the warm-up stage. A detailed analysis of this effect will be the subject of future work.

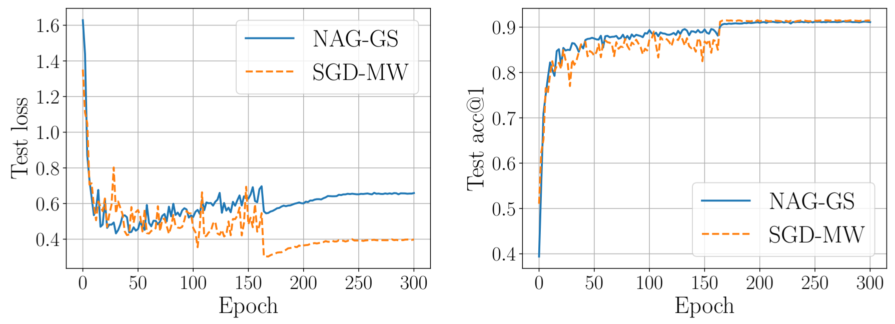


Fig. 5: Comparison of NAG-GS and SGD-MW on ResNet-20 model and CIFAR-10. NAG-GS outperforms SGD-MW uniformly in the first 150 epochs and provides the same accuracy further.

3.4 Transformer Models

3.4.1 RoBERTa

In this section, we test NAG-GS for fine-tuning the pre-trained RoBERTa model from TRANSFORMERS library [29] on GLUE benchmark datasets [26]. In this benchmark, the reference optimizer is AdamW [16] with a polynomial learning rate schedule. The training setup defined in [15] is used for both NAG-GS and AdamW optimizers.

We search for an optimal learning rate and γ for the NAG-GS optimizer with fixed $\mu = 1$ to get the best performance on the task. Search space consists of learning rate α from $[10^{-3},\ 10^{0}]$, factor γ from $[10^{-2},\ 10^{0}]$. Note that NAG-GS is used with a constant learning rate to simplify hyperparameter search.

Regarding learning rate values, the one allowed by AdamW is around 10^{-5} while NAG-GS allows a much larger value of 10^{-2}. At the same time, optimal value for γ in NAG-GS is found to be 1. Evaluation results on GLUE tasks are presented in Table 3. Despite a rather restrained search space for NAG-GS hyperparameters, it demonstrates better performance on some tasks and competitive performance on others.

Table 3: Comparison of AdamW and NAG-GS optimizers in fine-tuning on GLUE benchmark. We use reported hyperparameters for AdamW. We search hyperparameters of NAG-GS for the best performance metric. NAG-GS outperforms or is competitive with AdamW on the considered tasks.

Optimizer	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST2	STS-B	WNLI
AdamW	**61.60**	**87.56**	88.24	**92.62**	**91.69**	**78.34**	**94.95**	**90.68**	**56.34**
NAG-GS	**61.60**	87.24	**90.69**	92.59	91.01	77.97	94.50	90.21	**56.34**

3.4.2 Vision Transformer model

We used the Vision Transformer model [30], which was pre-trained on the ImageNet dataset [6], and fine-tuned it on the `food101` dataset [4] using NAG-GS and AdamW. This task involves classifying a dataset of 101 food categories, with 1000 images per class. To ensure a fair comparison, we first conducted an intensive hyperparameter search [3] for all possible hyperparameter configurations on a subset of the data for each method and selected the best configuration. The intervals of the search are the following: $\text{lr} \sim \text{LogUniform}(10^{-4}, 10^{-1})$, $\mu \sim \text{Uniform}(10^{-2}, 10)$, $\gamma \sim \text{Uniform}(10^{-2}, 10)$. After the hyperparameter search, we performed the experiments on the entire dataset. The results are presented in Table 4. We observed that properly-tuned NAG-GS outperformed AdamW in both training and evaluation metrics. Also, NAG-GS reached higher accuracy than AdamW after one epoch. The optimal hyperparameters found for NAG-GS are $\alpha = 0.07929, \gamma = 0.3554, \mu = 0.1301$; for AdamW $\text{lr} = 0.00004949, \beta_1 = 0.8679, \beta_2 = 0.9969$.

Table 4: Test accuracies of NAG-GS and AdamW for Vision Transformer model fine-tuned on `food101` dataset. The NAG-GS outperforms AdamW after the presented number of epochs.

Stage	NAG-GS	AdamW
After 1 epoch	**0.8419**	0.8269
After 25 epochs	**0.8606**	0.8324

4 Conclusion and further works

We have presented a new and theoretically motivated stochastic optimizer called NAG-GS. It comes from the semi-implicit Gauss-Seidel discretization of a well-chosen accelerated Nesterov-like SDE. These building blocks ensure two central properties for NAG-GS: (1) the ability to accelerate the optimization process and (2) better robustness to large learning rates. We demonstrate these features theoretically and provide a detailed analysis of the convergence of the method in the quadratic case. Moreover, we show that NAG-GS is competitive with state-of-the-art methods for training a small logistic regression model and larger models like ResNet-20, VGG-11, and Transformers. In numerical tests, NAG-GS demonstrates faster convergence in the first epochs due to the larger learning rate and similar final scores to standard optimizers, which work only with smaller learning rates. Further work will focus on the non-asymptotic convergence analysis of NAG-GS and the development of proper learning rate schedulers for it.

Acknowledgments

VL acknowledges the support by the Ministry of Economic Development of the Russian Federation (agreement No. 139-10-2025-034 dd. 19.06.2025, IGK 000000C313925P4D0002).

References

1. Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M.: Optuna: A next-generation hyperparameter optimization framework. In: Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2019)
2. Attouch, H., Chbani, Z., Riahi, H.: Fast proximal methods via time scaling of damped inertial dynamics. SIAM Journal on Optimization **29**(3), 2227–2256 (2019)
3. Biewald, L.: Experiment tracking with weights and biases (2020). URL `https://www.wandb.com/`. Software available from wandb.com
4. Bossard, L., Guillaumin, M., Van Gool, L.: Food-101 – mining discriminative components with random forests. In: European Conference on Computer Vision (2014)
5. Dahlquist, G.G.: A special stability problem for linear multistep methods. BIT Numerical Mathematics **3**(1), 27–43 (1963)
6. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 248–255 (2009). DOI 10.1109/CVPR.2009.5206848
7. Franca, G., Robinson, D., Vidal, R.: Admm and accelerated admm as continuous dynamical systems. In: International Conference on Machine Learning, pp. 1559–1567. PMLR (2018)

8. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778 (2016)
9. Keskar, N.S., Mudigere, D., Nocedal, J., Smelyanskiy, M., Tang, P.T.P.: On large-batch training for deep learning: Generalization gap and sharp minima. In: International Conference on Learning Representations (2017). URL `https://openreview.net/forum?id=H1oyRlYgg`
10. Krichene, W., Bayen, A., Bartlett, P.L.: Accelerated mirror descent in continuous and discrete time. Advances in neural information processing systems **28** (2015)
11. Krizhevsky, A.: Learning multiple layers of features from tiny images. Master's thesis, University of Toronto (2009)
12. Laborde, M., Oberman, A.: A lyapunov analysis for accelerated gradient methods: from deterministic to stochastic case. In: International Conference on Artificial Intelligence and Statistics, pp. 602–612. PMLR (2020)
13. Lan, G.: An optimal method for stochastic composite optimization. Mathematical Programming **133**(1), 365–397 (2012). URL `https://doi.org/10.1007/s10107-010-0434-y`
14. LeCun, Y., Cortes, C., Burges, C.: Mnist handwritten digit database. ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist **2** (2010)
15. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692 (2019)
16. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: International Conference on Learning Representations (2019)
17. Luo, H., Chen, L.: From differential equation solvers to accelerated first-order methods for convex optimization. Mathematical Programming pp. 1–47 (2021)
18. Malladi, S., Lyu, K., Panigrahi, A., Arora, S.: On the sdes and scaling rules for adaptive gradient algorithms. arXiv preprint arXiv:2205.10287 (2022)
19. Muehlebach, M., Jordan, M.: A dynamical systems perspective on nesterov acceleration. In: International Conference on Machine Learning, pp. 4656–4662. PMLR (2019)
20. Nesterov, Y.: Lectures on Convex optimization, vol. 137. Springer Optimization and Its Applications (2018)
21. Shi, B., Du, S.S., Jordan, M.I., Su, W.J.: Understanding the acceleration phenomenon via high-resolution differential equations. Mathematical Programming pp. 1–70 (2021)
22. Shi, B., Du, S.S., Su, W., Jordan, M.I.: Acceleration via symplectic discretization of high-resolution differential equations. Advances in Neural Information Processing Systems **32** (2019)
23. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
24. Su, W., Boyd, S., Candes, E.: A differential equation for modeling nesterov's accelerated gradient method: theory and insights. Advances in neural information processing systems **27** (2014)
25. Taylor, A., Drori, Y.: An optimal gradient method for smooth strongly convex minimization. Mathematical Programming **199**(1-2), 557–594 (2023)
26. Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., Bowman, S.R.: Glue: A multi-task benchmark and analysis platform for natural language understanding. arXiv preprint arXiv:1804.07461 (2018)
27. Wilson, A.C., Recht, B., Jordan, M.I.: A Lyapunov Analysis of Momentum Methods in Optimization (2018). URL `https://arxiv.org/abs/1611.02635`
28. Wilson, A.C., Recht, B., Jordan, M.I.: A Lyapunov Analysis of Accelerated Methods in Optimization. Journal of Machine Learning Research **22**(113), 1–34 (2021). URL `http://jmlr.org/papers/v22/20-195.html`
29. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T.L., Gugger, S., Drame, M., Lhoest, Q., Rush, A.M.: Transformers: State-of-the-art natural language processing. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pp. 38–45. Association for Computational Linguistics, Online (2020). URL `https://www.aclweb.org/anthology/2020.emnlp-demos.6`
30. Wu, B., Xu, C., Dai, X., Wan, A., Zhang, P., Yan, Z., Tomizuka, M., Gonzalez, J., Keutzer, K., Vajda, P.: Visual transformers: Token-based image representation and processing for computer vision (2020)
31. Zhang, J., Mokhtari, A., Sra, S., Jadbabaie, A.: Direct runge-kutta discretization achieves acceleration. Advances in neural information processing systems **31** (2018)
32. Zhou, K., So, A.M.C., Cheng, J.: Boosting first-order methods by shifting objective: new schemes with faster worst-case rates. Advances in Neural Information Processing Systems **33**, 15,405–15,416 (2020)