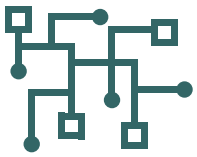

ISE

Industrial and
Systems Engineering

Fast and Safe: Accelerated Gradient Methods With Optimality Certificates And Underestimate Sequences

MAJID JAHANI[1], NAGA VENKATA C. GUDAPATI[1], CHENXIN MA[1], RACHAEL TAPPENDEN[2], AND MARTIN TAKÁČ[1]

[1]Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA, USA

[2]School of Mathematics and Statistics, University of Canterbury, Private Bag 4800, Christchurch 8140, New Zealand

ISE Technical Report 21T-003

LEHIGH
UNIVERSITY

Fast and Safe: Accelerated gradient methods with optimality certificates and underestimate sequences

Majid Jahani · Naga Venkata C. Gudapati · Chenxin Ma · Rachael Tappenden · Martin Takáč

Received: date / Accepted: date

Abstract In this work we introduce the concept of an Underestimate Sequence (UES), which is motivated by Nesterov's estimate sequence. Our definition of a UES utilizes three sequences, one of which is a lower bound (or under-estimator) of the objective function. The question of how to construct an appropriate sequence of lower bounds is addressed, and we present lower bounds for strongly convex smooth functions and for strongly convex composite functions, which adhere to the UES framework. Further, we propose several first order methods for minimizing strongly convex functions in both the smooth and composite cases. The algorithms, based on e□ciently updating lower bounds on the objective functions, have natural stopping conditions that provide the user with a certificate of optimality. Convergence of all algorithms is guaranteed through the UES framework, and we show that all presented algorithms converge linearly, with the accelerated variants enjoying the optimal linear rate of convergence.

Keywords Underestimate Sequence · Estimate Sequence · Quadratic Averaging · Lower bounds · Strongly convex · Smooth minimization · Composite minimization · Accelerated Algorithms

Mathematics Subject Classification (2010) 90C25 · 90C47 · 68Q25

M. Jahani, N. V. C. Gudapati, C. Ma, M. Takáč
Department of Systems and Industrial Engineering, Lehigh University, H.S. Mohler Laboratory, 200 West Packer Avenue, Bethlehem, PA 18015, USA.
E-mail: maj316@lehigh.edu,nag415@lehigh.edu, chm514@lehigh.edu, Takac.MT@gmail.com

R. Tappenden
School of Mathematics and Statistics, University of Canterbury, Private Bag 4800, Christchurch 8140, New Zealand. E-mail: rachael.tappenden@canterbury.ac.nz

Martin Takáč was supported by NSF Grants CCF-1618717 and CMMI-1663256.

1 Introduction

In this work we are interested in solving the strongly convex, composite, optimization problem,

$$\min_{x \in \mathbb{R}^n} \{F(x) := f(x) + h(x)\}. \tag{1}$$

We use x^* to denote the optimal solution of (1), and $F^* := F(x^*)$ to denote the associated optimal function value. It is assumed that h is a convex and possibly nonsmooth function.

Assumption 1 *The function $f(\cdot)$ is μ-strongly convex and L-smooth, i.e., for all $x, y \in \mathbb{R}^n$, it holds that*

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

$$f(x) \leq f(y) + \langle \nabla f(y), x - y \rangle + \frac{L}{2} \|x - y\|^2. \tag{3}$$

In words, Assumption 1 explains that f is assumed to be continuously differentiable on $\mathbb{R}^n$ with $L \geq \mu > 0$. It is straightforward to show that strong convexity of f implies strong convexity of F.

For problems of the form (1), which satisfy Assumption 1, it is well known that Nesterov's methods [21, 23, 25] converge linearly, with the accelerated variants converging at the optimal rate of $(1 - \sqrt{\mu/L})$.

Nesterov's acceleration approach, and the idea of adding momentum, has led to the extensive analysis of accelerated first order methods in a variety of settings. This includes a recent surge of interest in investigating stochastic gradient methods [11, 13, 29, 30] and their accelerated variants [6, 14, 26, 31]. Coordinate descent methods [10, 19, 28, 33] are another class of algorithms that have proved extremely popular, largely because they can take advantage of modern parallel computing architecture [12, 18], and this has also inspired much research into studying their accelerated versions [1, 9, 17]. However, while the theoretical and practical performance of Nesterov's methods is well established, a satisfactory geometric interpretation of these approaches has been elusive.

Recently the authors of [3,8] proposed algorithms for smooth functions (i.e., $h \equiv 0$ in (1)) that enjoy the same optimal rate of convergence as Nesterov's accelerated method, but also have a novel geometric intuition. Specifically, the geometric descent algorithm [3] achieves the optimal linear convergence rate, and shares a geometric intuition similar to that of ellipsoidal methods. The authors illustrate that the optimal rate is obtained by appropriately shrinking two balls that contain x^* (the minimizer of $f(x)$) at each iteration.

Motivated by [3], the paper [8] proposed the Optimal Quadratic Averaging (OQA) algorithm. Indeed, [8] show that their OQA algorithm generates the same iterate sequence as that generated by the algorithm in [3], although the two schemes are slightly different. The OQA algorithm also maintains a sequence of quadratic lower bounds on the objective function, and at each iteration the new quadratic lower bound is formed as the optimal average of the current lower bound and the lower bound from the previous iteration. The

gap between the function value $f(x_k)$ and the minimum value of lower bound, say φ_k, converges to zero at the optimal rate. Importantly, the lower bound also acts a natural stopping criterion for the algorithm, and when $f(x_k) - \varphi_k \leq$, where > 0 is some stopping tolerance, then the user has a certificate of - optimality, i.e., it is guaranteed that $f(x_k) - f \leq$. In practice, the OQA algorithm can be equipped with historical information to achieve further speed up. However, the OQA algorithm and its history based variant need at least two calls of a line search process at every iteration, which can pose a heavy computational burden in terms of function evaluations. The authors in [8] also briefly describe how their *unaccelerated OQA algorithm* can be extended to composite functions, and left as an open problem the possibility of deriving *accelerated* proximal variants.

More recently, the authors of [5] successfully addressed the open problem in [8] and presented an accelerated algorithm for composite problems of the form (1), that achieves the optimal linear rate of convergence. Their algorithm, called the geometric proximal gradient (GeoPG) method also has a satisfying geometrical interpretation similar to that in [3]. Unfortunately, a major drawback of GeoPG in [5] is that the algorithm is rather complicated, and requires a couple of inner loops to determine necessary algorithm parameters. For example, for GeoPG one must find the root of a specific function and one is also required to compute a minimum enclosing ball via some iterative process; both of these steps must be carried out at every iteration, which is expensive.

Another relevant work is that in [11] where the authors propose an accelerated stochastic approximation algorithm. That algorithm is based upon a modification of Nesterov's optimal smooth method [20] to fit a convex composite setting where only a noisy gradient is available. Section 5 of [11] describes how certain stochastic lower bound can be incorporated into their method, although some additional computational e□ort is required to compute these. Furthermore, in [7] the authors provide a general scheme for the analysis of first-order methods by construction of a 'duality gap' involving an approximation of the objective function. In the continuous-time setting, the authors in [7] show that the approximate duality gap decreases. They also characterize the discretization errors incurred by di□erent discretization methods (please see [7] for more details).

In this paper we propose several new algorithms to solve problem (1) that are motivated by, and extend, the previously mentioned works. In particular, we present four algorithms: a proximal Gradient Descent (GD) type algorithm for composite problems, an accelerated proximal GD type algorithm for composite problems, a GD type algorithm for smooth problems, and an accelerated GD type algorithm for smooth problems. Our algorithms all converge linearly, and the accelerated variants converge at the optimal linear rate. These algorithms blend the positive features of Nesterov's methods [21, 23, 25] and the OQA algorithm [8], and thus enjoy the advantages of both approaches. First, similarly to Nesterov's methods, no line search is needed by any of our algorithms as long as we make the standard assumption that the Lipschitz constant L is known or is easily computable. Hence, there are no 'inner-loops'

in any of our algorithm variants, which ensures that the computational cost is low and is fixed at every iteration. Secondly, our algorithms incorporate quadratic lower bounds so they have natural stopping conditions; a feature that is similar to OQA. Each iteration of the OQA method in [8] requires two 'optimal steps': an optimal line-search in a given direction and the optimal choice of the averaging of successive quadratics. In this work, we show that instead of making such optimal choices, one can use carefully coupled convex combinations in the two steps, and this approach is computationally cheaper. The resulting scheme maintains the optimal linear rate of convergence and naturally extends to the proximal setting.

Furthermore, in this work we propose the definition of an UnderEstimate Sequence (UES), which was motivated by Nesterov's Estimate Sequence [20]. Perhaps surprisingly, early work on estimate sequences appeared to be largely overlooked, but since Nesterov's work on smoothing techniques in the early 2000s [22], they have seen a revival in popularity. For example, the work of Baes in [2], the development of a randomized estimate sequence in [16] and an approximate estimate sequence in [15]. The definition we introduce is different from previous works because an underestimate sequence involves *lower bounds* on the objective function. To the best of our knowledge, this is the first work which proposes estimate sequences that form *lower bounds* on the objective function. The UES definition is the powerhouse of our convergence analysis; we prove that each of our proposed algorithms generates a UES, and consequently the algorithms converge (linearly) to the optimal solution of problem (1). While we describe 4 new algorithms in this work, we stress that the UES definition is general, and it allows a plethora of algorithms to be developed. Moreover, any developed algorithm whose iterates generate a UES is guaranteed to converge to the optimal value F (under a mild assumption on one of the algorithm parameters).

The algorithms presented in this work can be used to solve problem (1). However, they are also more widely applicable as a subproblem solver in existing optimization algorithms. In particular, the Inexact Coordinate Descent algorithm [32], and the Universal Catalyst algorithm [15], are guaranteed to converge if at every iteration, the inexact solution to a certain subproblem is -optimal. In both cases, the algorithms solve optimization problems where the objective function is convex (but not necessarily strongly convex) *but the arising subproblems are strongly convex*. Neither paper explains how to verify the subproblem stopping conditions. This work builds the link, bridging theory and practice, by providing algorithms that can be used to solve subproblems and which return solutions that are easily verified to be -optimal.

1.1 Contributions

The main contributions of this paper are stated now (listed in no particular order).

- Underestimate Sequence. We introduce the concept of an UnderEstimate Sequence (UES).The UES consists of three sequences $\{x_k\}_{k=0}^{\infty}$, $\{\phi_k(x)\}_{k=0}^{\infty}$ and $\{\alpha_k\}_{k=0}^{\infty}$, where for all k, $\phi_k(x)$ is a *global lower bound* on the objective function $F(x)$. While there have been several extensions and variants of Nesterov's work on estimate sequences [20], the definition of a UES involves *lower bounds* or *under*estimates of $F(x)$, which is new. The UES framework is general, conceptually simple, and it allows the construction of a wide variety of algorithms to solve (1).
- New algorithms. Four new algorithms are presented that are computationally efficient and whose iterates generate a UES. Crucially, two of our algorithms solve the *composite* problem (1). The algorithms are: (i) CUESA, a proximal GD type algorithm for composite problems; (ii) ACUESA, an accelerated proximal GD type algorithm for composite problems; (iii) SUESA, a GD type algorithm for smooth problems; and (iv) ASUESA, an accelerated GD type algorithm for smooth problems.
- Algorithms with optimal convergence rate. Each of the four algorithms generate iterates that form a UES, and all are guaranteed to converge to the optimal solution of (1). Moreover, all algorithms converge linearly, and the accelerated algorithms (ACUESA and ASUESA) converge *at the optimal rate.*
- Algorithms with convergence certificates. The underestimate sequence builds a global lower bound of $F(x)$ at each iteration, and the gap between the (minimum of the) lower bound and $F(x_k)$ tends to zero. Thus, this difference acts as a kind of surrogate "duality gap", and once this gap falls below some (user defined) stopping tolerance , it is guaranteed that the point returned by the algorithm is -optimal. The algorithms can be used as subproblem solvers within existing algorithms, to ensure that the solutions to the subproblems are -accurate.
- No line search. The algorithms developed in this work are computationally efficient and do not involve any 'inner loops'. In contrast, the methods in [3, 5, 8] all involve an exact linesearch or a root finding process to determine necessary algorithmic parameters, which comes with an additional computational cost.

1.2 Paper Outline

The paper is organized as follows. In the next section the definition of an Underestimate Sequence (UES) is presented, along with a proposition which shows that if one has a UES, then, under a mild assumption, it is guaranteed that $F(x_k) - F \to 0$ linearly. Section 3 discusses lower bounds for function F (in both the smooth and composite cases), and these lower bounds are a critical part of the underestimate sequences framework. In Section 4 we propose two algorithms for solving (1) in the composite case ($h = 0$) and in Section 5 we present two algorithms for solving (1) in the smooth case ($h \equiv 0$). The algorithms are supported by convergence theory, which shows that they are

all guaranteed to converge linearly, and the accelerated algorithms achieve the optimal rate. In Section 6 an adaptive Lipschitz constant updating scheme is presented that can be used as an inner loop within the four previously mentioned algorithms, so that the true Lipschitz constant is not explicitly needed. Section 7 presents numerical experiments to demonstrate the practical advantages of our proposed algorithms, and concluding remarks are given in Section 8.

2 Underestimate Sequence

In this section, the definition of an Underestimate Sequence (UES) is presented, as well as a proposition showing that if one has a UES then $F(x_k) - F^* \to 0$.

Definition 1 A series of sequences $\{x_k\}_{k=0}^{\infty}$, $\{\varphi_k(x)\}_{k=0}^{\infty}$ and $\{\alpha_k\}_{k=0}^{\infty}$, where $\alpha_k \in (0,1)$ for all $k \geq 0$, is called an Underestimate Sequence (UES) of the function $F(x)$ if, for all $x \in \mathbb{R}^n$ and for all $k \geq 0$ we have,

$$\varphi_k(x) \leq F(x), \tag{4}$$

$$F(x_{k+1}) - \varphi_{k+1} \leq (1-\alpha_k)(F(x_k) - \varphi_k), \tag{5}$$

where $\varphi_k^* := \min_x \varphi_k(x)$.

Proposition 1 *If $\{x_k\}_{k=0}^{\infty}$, $\{\varphi_k(x)\}_{k=0}^{\infty}$ and $\{\alpha_k\}_{k=0}^{\infty}$ is a UES of $F(x)$, then*

$$F(x_k) - \varphi_k^* \leq \lambda_k(F(x_0) - \varphi_0^*), \tag{6}$$

where $\lambda_0 = 1$ and $\lambda_{k+1} = (1-\alpha_k)\lambda_k$. Furthermore, since $\varphi_k^* \leq \varphi_k(x^*) \leq F^*$ for all $k \geq 0$, if $\lambda_k \to 0$, then (6) *implies that $\{F(x_k) - F^*\}_{k=0}^{\infty}$ converges to* 0.

Remark 1 Note that if $\prod_{k=0}^{\infty} \alpha_k = \infty$ then $\lambda_k \to 0$; also see [21, pg.72–73].

Definition 1 is different from Nesterov's Estimate Sequence (ES) in several ways. Although both a UES and an ES contain a sequence of estimators $\{\varphi_k(x)\}_{k=0}^{\infty}$ for F, Definition 1 shows that φ_k must be a *lower/under estimator* of F for all $k \geq 0$, but this necessarily does not hold for an ES. Nesterov's ES convergence guarantees rely upon $F(x_k) \leq \varphi_k$ holding, but this *does not hold* in our case. Moreover, the definition of an ES only contains two sequences, while the UES has an extra sequence of points $\{x_k\}_{k=0}^{\infty}$. This enables us to show that the gap between the function value at x_k and φ_k decreases in the kth iteration. The quantities in Definition 1 and Proposition 1 are all computable; they do not require knowledge of x^*. (A more detailed comparison can be found in Appendix A.)

Proposition 1 shows that, if $\lambda_k \to 0$ (where $\lambda_{k+1} = \prod_{j=0}^{k}(1-\alpha_j)$), then any sequences that form a UES (i.e., satisfy Definition 1) are guaranteed to

converge to the optimal solution of problem (1), and $F(x_k) - \varphi_k \to 0$. Additionally, if $\alpha_k \in (0, 1)$ is constant for all $k \geq 0$, then a linear rate of convergence holds. Thus, the UES construction provides a general framework for determining whether an optimization algorithm for problem (1) will converge (linearly). In particular, if the iterates generated by an optimization algorithm satisfy Definition 1 and $\lambda_k \to 0$, then that algorithm converges.

The UES framework is not only interesting from a theoretical perspective, but it also has an important practical advantage. In particular, $F(x_k) - \varphi_k$ provides a natural stopping criterion when designing algorithms, because $F(x_k)$ and φ_k are upper and lower bounds for F^*, respectively. This difference is a kind of surrogate for the duality gap, so algorithms that adhere to the UES framework are provided with a certificate of optimality, which is a desirable attribute.

3 Lower Bounds via Quadratic Averaging

The purpose of this section is to introduce (global) lower bounds for the function F defined in (1), in both the smooth ($h = 0$) and nonsmooth cases. Lower bounds are the cornerstone of the UES set up, as seen in (4) in Definition 1. The efficient construction of global lower bounds for F allows the development of practical algorithms whose convergence can be analyzed via the UES framework.

Before stating the lower bounds, several technical results are presented that will be used throughout this paper.

3.1 Preliminary Technical Results

The proximal map is defined as

$$\text{prox}_{h,\gamma}(x) := \arg\min_u \{h(u) + \tfrac{\gamma}{2}\|x - u\|^2\}, \tag{7}$$

and the proximal gradient is

$$G_\gamma(x) := \gamma\left(x - \text{prox}_{h,\gamma}\left(x - \tfrac{1}{\gamma}\nabla f(x)\right)\right). \tag{8}$$

Definitions (7) and (8) will be used with $\gamma \equiv L$. Given some point $x \in \mathbb{R}^n$, a short step and a long step are denoted by

$$x^+ := x - \tfrac{1}{L} G_L(x), \tag{9}$$

$$x^{++} := x - \tfrac{1}{\mu} G_L(x). \tag{10}$$

In the smooth case ($h \equiv 0$), the proximal gradient is simply the gradient $\nabla f(\cdot)$, so the short and long steps ((9) and (10)) simplify as

$$x^+ = x - \tfrac{1}{L}\nabla f(x) \tag{11}$$

$$x^{++} = x - \tfrac{1}{\mu}\nabla f(x). \tag{12}$$

For a function $h : \mathrm{R}^n \to \mathrm{R}$, the elements $s \in \mathrm{R}^n$ that satisfy

$$h(y) \geq h(x) + \langle s, y - x \rangle, \qquad y \in \mathrm{R}^n,$$

are called the subgradients of h at the point x. In words, all elements defining a linear function that supports h at a point x are subgradients. The set of all s at a point x is called the subdifferential of h and it is denoted by $\partial h(x)$. The following Lemma characterizes elements of $\partial h(x^+)$.

Lemma 1 *Let $G_L(x)$ and x^+ be defined in* (8) *and* (9)*, respectively. Then, for all $x \in \mathrm{R}^n$, $G_L(x) - \nabla f(x) \in \partial h(x^+)$.*

Proof For a given point $x \in \mathrm{R}^n$, combining (7), (8) and (9) gives

$$x^+ = \arg\min_u \left\{ h(u) + \tfrac{L}{2} \| u - (x - \tfrac{1}{L} \nabla f(x)) \|^2 \right\}.$$

Then

$$0 \in \tfrac{1}{L}\partial h(x^+) + x^+ - (x - \tfrac{1}{L} \nabla f(x)) \overset{(9)}{=} \tfrac{1}{L}\partial h(x^+) - \tfrac{1}{L}(G_L(x) - \nabla f(x)).$$

Multiplying through by L, and rearranging, gives the result.

3.2 A Lower Bound for Composite Functions

Here, a lower bound is developed for composite function, i.e., it is assumed that h is not equivalent to the zero function. The following Lemma defines a lower bound for $F(x)$ in (1). The lower bound is the same as that presented in [5] and [8], with the roles of x and y reversed here; the proof is included for completeness.

Lemma 2 (Lemma 3.1 in [5]; Lemma 6.1 in [8]) *Given a point $y \in \mathrm{R}^n$, let $G_L(y)$ and y^+ be defined in* (8) *and* (9)*, respectively. Then for all $x \in \mathrm{R}^n$*

$$\phi(x; y) := F(y^+) + \langle G_L(y), x - y \rangle + \tfrac{\mu}{2} \| x - y \|^2 + \tfrac{1}{2L} \| G_L(y) \|^2 \leq F(x). \quad (13)$$

Proof By Assumption 1 (μ-strongly convex)

$$f(y) + \langle \nabla f(y), x - y \rangle + \tfrac{\mu}{2} \| x - y \|^2 \leq f(x), \qquad x, y \in \mathrm{R}^n, \quad (14)$$

and (L-smooth)

$$\begin{aligned} f(y^+) &\leq f(y) + \langle \nabla f(y), y^+ - y \rangle + \tfrac{L}{2} \| y^+ - y \|^2 \\ &\overset{(9)}{=} f(y) - \tfrac{1}{L} \langle \nabla f(y), G_L(y) \rangle + \tfrac{1}{2L} \| G_L(y) \|^2. \end{aligned} \quad (15)$$

Combining (14) and (15) gives

$$\begin{aligned} F(y^+) &\leq F(x) - \langle f(y), x - y\rangle - \tfrac{\mu}{2}\|x - y\|^2 - \tfrac{1}{L}\langle f(y), G_L(y)\rangle \\ &\quad + \tfrac{1}{2L}\|G_L(y)\|^2 + (h(y^+) - h(x)) \\ &= F(x) - \langle f(y), x - y^+\rangle - \tfrac{\mu}{2}\|x - y\|^2 \\ &\quad + \tfrac{1}{2L}\|G_L(y)\|^2 + (h(y^+) - h(x)) \\ &= F(x) - \langle f(y) - G_L(y), x - y^+\rangle - \tfrac{\mu}{2}\|x - y\|^2 \\ &\quad + \tfrac{1}{2L}\|G_L(y)\|^2 + (h(y^+) - h(x)) - \langle G_L(y), x - y^+\rangle \\ &\overset{\text{Lemma 1}}{\leq} F(x) - \tfrac{\mu}{2}\|x - y\|^2 + \tfrac{1}{2L}\|G_L(y)\|^2 - \langle G_L(y), x - y^+\rangle \\ &= F(x) - \tfrac{\mu}{2}\|x - y\|^2 - \tfrac{1}{2L}\|G_L(y)\|^2 - \langle G_L(y), x - y\rangle . \end{aligned}$$

Rearranging gives the result.

Before stating the next result, which shows that $\phi(x;y)$ is a quadratic lower bound, we give the following equivalence,

$$\tfrac{\mu}{2}\|x - y^{++}\|^2 \overset{(10)}{=} \tfrac{\mu}{2}\|x - y\|^2 + \langle x - y, G_L(y)\rangle + \tfrac{1}{2\mu}\|G_L(y)\|^2. \tag{16}$$

Lemma 3 *For all $x, y \in \mathbb{R}^n$, the lower bound* (13) *has the canonical form*

$$\phi(x;y) = \phi^* + \tfrac{\mu}{2}\|x - y^{++}\|^2, \tag{17}$$

where

$$\phi^* = F(y^+) + \left(\tfrac{1}{2L} - \tfrac{1}{2\mu}\right)\|G_L(y)\|^2. \tag{18}$$

Proof Minimizing $\phi(x;y)$ in (13) w.r.t. x, and using the definition in (7), yields the minimizer

$$y^{++} = \arg\min_x \phi(x;y). \tag{19}$$

The corresponding minimal value is

$$\begin{aligned} \phi^* &:= \min_x \phi(x;y) = \phi(y^{++};y) \\ &\overset{(13)}{=} F(y^+) + \langle G_L(y), y^{++} - y\rangle + \tfrac{\mu}{2}\|y^{++} - y\|^2 + \tfrac{1}{2L}\|G_L(y)\|^2 \\ &\overset{(10)}{=} F(y^+) - \tfrac{1}{\mu}\langle G_L(y), G_L(y)\rangle + \tfrac{\mu}{2}\tfrac{1}{\mu^2}\|G_L(y)\|^2 + \tfrac{1}{2L}\|G_L(y)\|^2 \\ &= F(y^+) + \left(\tfrac{1}{2L} - \tfrac{1}{2\mu}\right)\|G_L(y)\|^2, \end{aligned} \tag{20}$$

which is equivalent to (19). (Note also that (19) and (18) are the minimizer and minimum value of (17), respectively.) Furthermore,

$$\begin{aligned} \phi(x;y) &\overset{(13)}{=} F(y^+) + \langle G_L(y), x - y\rangle + \tfrac{\mu}{2}\|x - y\|^2 + \tfrac{1}{2L}\|G_L(y)\|^2 \\ &\overset{(16)}{=} F(y^+) + \left(\tfrac{1}{2L} - \tfrac{1}{2\mu}\right)\|G_L(y)\|^2 + \tfrac{\mu}{2}\|x - y^{++}\|^2, \end{aligned}$$

which, by (20), confirms that (17) is equivalent to (13).

Remark 2 Lemma 3 shows that the lower bound (13) (equivalently (17)) is a quadratic lower bound for $F(x)$.

Now, a sequence of lower bounds $\{\phi_k(x)\}_{k=0}^{\infty}$ can be defined in the following way. Using (13) and a given initial point x_0, define the function

$$\phi_0(x) := \phi(x; x_0) = \phi_0 + \tfrac{\mu}{2} \|x - v_0\|^2, \tag{21}$$

where

$$\phi_0 := F(x_0^+) + \left(\tfrac{1}{2L} - \tfrac{1}{2\mu}\right) \|G_L(x_0)\|^2 \quad \text{and} \quad v_0 = x_0^{++}. \tag{22}$$

Differentiating (21) w.r.t. x shows that the minimum value and minimizer of $\phi_0(x)$ are given by (22). This motivates the following construction: for a given $x_0 \in \mathbb{R}^n$,

1. $\phi_0(x) := \phi(x; x_0) = \phi_0 + \tfrac{\mu}{2} \|x - v_0\|^2$,
2. For $k \geq 0$, $\alpha_k \in (0, 1)$, and some point y_k, recursively define

$$\phi_{k+1}(x) := (1 - \alpha_k)\phi_k(x) + \alpha_k \phi(x; y_k). \tag{23}$$

Lemma 4 *For all $k \geq 0$, ϕ_{k+1} can be written in the canonical form*

$$\phi_{k+1}(x) := \phi_{k+1} + \tfrac{\mu}{2} \|x - v_{k+1}\|^2, \tag{24}$$

where

$$v_{k+1} := (1 - \alpha_k)v_k + \alpha_k y_k^{++} \tag{25}$$

$$\begin{aligned} \phi_{k+1} := {} & (1 - \alpha_k)\left(\phi_k + \tfrac{\mu}{2} \|v_{k+1} - v_k\|^2\right) \\ & + \alpha_k \left(F(y_k^+) + \left(\tfrac{1}{2L} - \tfrac{1}{2\mu}\right) \|G_L(y_k)\|^2 + \tfrac{\mu}{2} \|v_{k+1} - y_k^{++}\|^2\right). \end{aligned} \tag{26}$$

Proof The proof is by induction and the induction hypothesis is

$$\phi_k(x) = \phi_k + \tfrac{\mu}{2} \|x - v_k\|^2. \tag{27}$$

The following holds:

$$\begin{aligned} & (1 - \alpha_k) \|x - v_k\|^2 + \alpha_k \|x - y_k^{++}\|^2 \\ = {} & (1 - \alpha_k) \|x - v_{k+1} + v_{k+1} - v_k\|^2 + \alpha_k \|x - v_{k+1} + v_{k+1} - y_k^{++}\|^2 \\ = {} & (1 - \alpha_k) \left(\|x - v_{k+1}\|^2 + \|v_{k+1} - v_k\|^2 + 2\langle x - v_{k+1}, v_{k+1} - v_k\rangle\right) \\ & + \alpha_k \left(\|x - v_{k+1}\|^2 + \|v_{k+1} - y_k^{++}\|^2 + 2\langle x - v_{k+1}, v_{k+1} - y_k^{++}\rangle\right) \\ \overset{(25)}{=} {} & \|x - v_{k+1}\|^2 + (1 - \alpha_k) \|v_{k+1} - v_k\|^2 + \alpha_k \|x - v_{k+1}\|^2 \\ & + 2(1 - \alpha_k)\langle x - v_{k+1}, -\alpha_k(v_k - y_k^{++})\rangle + 2\alpha_k \langle x - v_{k+1}, (1 - \alpha_k)(v_k - y_k^{++})\rangle \\ = {} & \|x - v_{k+1}\|^2 + (1 - \alpha_k) \|v_{k+1} - v_k\|^2 + \alpha_k \|x - v_{k+1}\|^2. \end{aligned} \tag{28}$$

Now,

$$\begin{aligned}
\phi_{k+1}(x) &\overset{(23)}{=} (1-\alpha_k)\phi_k(x) + \alpha_k\phi(x;y_k) \\
&\overset{(27)}{=} (1-\alpha_k)(\phi_k + \tfrac{\mu}{2}\|x-v_k\|^2) + \alpha_k\phi(x;y_k) \\
&\overset{(17)+(18)}{=} (1-\alpha_k)(\phi_k + \tfrac{\mu}{2}\|x-v_k\|^2) \\
&\quad + \alpha_k\left[F(y_k^{++}) + \left(\tfrac{1}{2L} - \tfrac{1}{2\mu}\right)\|G_L(y_k)\|^2 + \tfrac{\mu}{2}\|x-y_k^{++}\|^2\right] \\
&\overset{(28)}{=} (1-\alpha_k)(\phi_k + \tfrac{\mu}{2}\|v_{k+1}-v_k\|^2) + \tfrac{\mu}{2}\|x-v_{k+1}\|^2 \\
&\quad + \alpha_k\left[F(y_k^{++}) + \left(\tfrac{1}{2L} - \tfrac{1}{2\mu}\right)\|G_L(y_k)\|^2 + \tfrac{\mu}{2}\|x-v_{k+1}\|^2\right] \\
&\overset{(26)}{=} \phi_{k+1} + \tfrac{\mu}{2}\|x-v_{k+1}\|^2.
\end{aligned}$$

The proof is complete.

Lemma 4 shows that the lower bound is quadratic. This feature is important because it is easy to find the minimizer and minimum value of a quadratic. Indeed, Lemma 4 shows that the minimizer of the quadratic lower bound $\varphi_{k+1}(x)$ is v_{k+1}, which is easily computed/updated using (25). Moreover, the minimum value of the quadratic is φ_{k+1} in (26), which is also readily available. Both v_{k+1} and φ_{k+1} are used in the algorithms presented later in this work, and the expressions given in the previous lemma show that they can be computed cheaply and only depend upon quantities that are computed at iteration k (no long memory is needed).

The following lemma gives an equivalent expression for ϕ_{k+1}, which is slightly more convenient in terms of ease of computation.

Lemma 5 *An equivalent expression for ϕ_{k+1} in* (26) *is*

$$\begin{aligned}
\phi_{k+1} = (1-\alpha_k)\left[\phi_k + \alpha_k\tfrac{\mu}{2}\|v_k - y_k^{++}\|^2\right] \\
+ \alpha_k\left[F(y_k^{++}) + \left(\tfrac{1}{2L} - \tfrac{1}{2\mu}\right)\|G_L(y_k)\|^2\right].
\end{aligned} \tag{29}$$

Proof Using (25) gives the equivalences

$$\|v_{k+1} - v_k\|^2 = \|(1-\alpha_k)v_k + \alpha_k y_k^{++} - v_k\|^2 = \alpha_k^2\|v_k - y_k^{++}\|^2 \tag{30}$$

and

$$\begin{aligned}
\|v_{k+1} - y_k^{++}\|^2 &= \|(1-\alpha_k)v_k + \alpha_k y_k^{++} - y_k^{++}\|^2 \\
&= (1-\alpha_k)^2\|v_k - y_k^{++}\|^2.
\end{aligned} \tag{31}$$

Combining (30) and (31) gives

$$\begin{aligned}
&(1-\alpha_k)\|v_{k+1} - v_k\|^2 + \alpha_k\|v_{k+1} - y_k^{++}\|^2 \\
&= \alpha_k^2(1-\alpha_k)\|v_k - y_k^{++}\|^2 + \alpha_k(1-\alpha_k)^2\|v_k - y_k^{++}\|^2 \\
&= \alpha_k(1-\alpha_k)(\alpha_k + (1-\alpha_k))\|v_k - y_k^{++}\|^2 \\
&= \alpha_k(1-\alpha_k)\|v_k - y_k^{++}\|^2.
\end{aligned} \tag{32}$$

Substituting (32) into (26) gives the result.

Lemma 6 *Let* $\alpha_k \in (0,1)$ *k. Then, for all* $x \in \mathrm{R}^n$, $\phi_k(x) \leq F(x)$ *for all* $k \geq 0$.

Proof The proof follows by a simple induction argument so is omitted for brevity.

Lemma 6 is fundamentally important for this work, because it shows that the quadratic bounds presented in the previous lemmas are *lower bounds* for $F(x_k)$ for all $k \geq 0$. If we can develop an algorithm that satisfies both conditions (4) and (5) in Definition (1), then the algorithm is guaranteed to converge by Proposition 1. But the quadratic lower bounds do indeed satisfy (4), so it remains to develop an algorithm whose iterates satisfy (5).

3.3 A Lower Bound for Smooth Functions

In this section, only smooth functions are considered, so here it is assumed that $h = 0$. Note that when $h \equiv 0$, (7) and (8) show that $G_L(y) = f(y)$. Now, for any point $y \in \mathrm{R}^n$, one can define a lower bound

$$\varphi(x;y) := f(y) - \tfrac{1}{2\mu} \| f(y) \|^2 + \tfrac{\mu}{2} \| x - y^{++} \|^2 \leq f(x), \tag{33}$$

which holds with equality $\varphi(x;y) = f(x)$ if and only if $x = y$. The lower bound in (33) is a consequence of Assumption 1 (μ-strongly convex) and the equivalence (16) used with $G_L(y) = f(y)$.

Remark 3 Notice that $\phi(x;y)$ in (13) does not generalize $\varphi(x;y)$ in (33) in the sense that (13) used with $h \equiv 0$ *does not* recover (33). (This will be discussed further in Section 5.3.)

Now, a sequence of lower bounds $\{\varphi_k(x)\}_{k=0}^{\infty}$ can be defined in the following way. Using (33) and a given initial point x_0, define the function

$$\varphi_0(x) := \varphi(x;x_0) = \varphi_0 + \tfrac{\mu}{2} \| x - v_0 \|^2, \tag{34}$$

where

$$\varphi_0 = f(x_0) - \tfrac{1}{2\mu} \| f(x_0) \|^2 \quad \text{and} \quad v_0 = x_0^{++}. \tag{35}$$

Differentiating the expression in (34) w.r.t. x shows that φ_0 and v_0 in (35) are the minimum value and minimizer of $\varphi_0(x)$, respectively. This motivates the following construction:

1. $\varphi_0(x) := \varphi(x;x_0) = \varphi_0 + \tfrac{\mu}{2} \| x - v_0 \|^2$
2. For $k \geq 0$, $\alpha_k \in (0,1)$, and some point y_k, recursively define

$$\varphi_{k+1}(x) := (1 - \alpha_k)\varphi_k(x) + \alpha_k \varphi(x;y_k). \tag{36}$$

Lemma 7 *For all $k \geq 0$, φ_{k+1} has the canonical form*

$$\varphi_{k+1}(x) = \varphi_{k+1}^{*} + \frac{\mu}{2} \|x - v_{k+1}\|^2, \tag{37}$$

where $\alpha_k \in (0,1)$, v_{k+1} is defined in (32) *and*

$$\varphi_{k+1}^{*} = (1-\alpha_k)\left(\varphi_k^{*} + \alpha_k \frac{\mu}{2} \|v_k - y_k^{++}\|^2\right) + \alpha_k\left(f(y_k) - \frac{1}{2\mu}\|\nabla f(y_k)\|^2\right). \tag{38}$$

The following Lemma shows that φ_k is a global lower bound for f.

Lemma 8 *Let $\alpha_k \in (0,1)$ k. Then, for all $x \in \mathbb{R}^n$, $\varphi_k(x) \leq F(x)$ for all $k \geq 0$.*

3.4 Geometry of the lower bounds

Here we reproduce the example in Section 2 of [8] to show how the lower bounds are combined to generate new quadratic lower bounds. All parameters used are identical to those in [8], but using the notation of this paper. The left plot shows the quadratics $\varphi_k(x) = 1 + 0.5(x+2)^2$ and $\varphi(x; x_k) = 3 + 0.5(x-4)^2$, and the resulting quadratics generated by taking convex combinations as described previously.

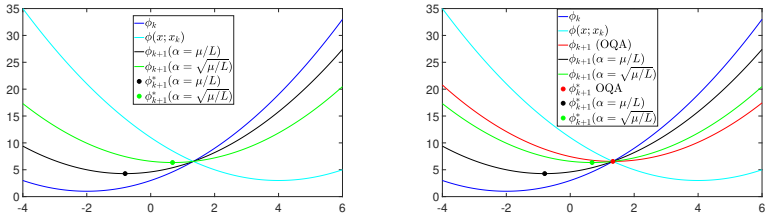

Fig. 1 Left: The quadratic lower bounds $\varphi_k(x) = 1 + 0.5(x+2)^2$ and $\varphi(x; x_k) = 3 + 0.5(x-4)^2$. The new lower bounds $\varphi_{k+1}(x)$ for $\alpha = \mu/L = 0.2$ (corresponding to the unaccelerated algorithm in this work) and $\varphi_{k+1}(x)$ for $\alpha = \sqrt{\mu/L} = \sqrt{0.2}$ (corresponding to the accelerated algorithm in this work). Right: This is the same plot as the left but it also shows the optimal quadratic average (corresponding to the OQA algorithm) from [8]. Finally, the minimizer of each quadratic model is indicated by the round marker.

4 Algorithms and Convergence Guarantees for Composite Functions

The purpose of this section is to demonstrate that the UES framework, and the previously presented lower bounds, are *useable* definitions that give rise to *efficient implementable algorithms*. Throughout this section we consider composite optimization problems (problems of the form (1) with $h = 0$) and, as for all results in this work, we suppose that Assumption 1 holds.

We present two algorithms whose iterates fit the Underestimate Sequence framework described in Section 2, and use the lower bounds developed in Section 3.2. Both algorithms fit the composite setting very naturally; the first algorithm is a proximal gradient descent type method, while the second algorithm is an accelerated proximal gradient variant. Both algorithms incorporate verifiable stopping conditions, and convergence guarantees are established via the UES framework.

4.1 A Composite Underestimate Sequence Algorithm

We now present an algorithm to solve (1), which is based on the UES framework. A brief description will follow.

Algorithm 1 Composite UES Algorithm (CUESA)

1: Initialization: Set $k = 0$, $\quad > 0$, initial point $x_0 \quad \mathrm{R}^n$ and compute μ, L.
2: Set v_0 and ϕ_0 as in (22) and let $\alpha_k = \frac{\mu}{L}$.
3: **while** $F(x_k) - \phi_k > \quad$ **do**
4: Set $y_k = x_k$, $y_k^+ = x_k^+$, and $y_k^{++} = x_k^{++}$
5: Set $x_{k+1} = x_k - \frac{1}{L} G_L(x_k)$,
6: Update v_{k+1} and ϕ_{k+1} as in (25) and (26) respectively.
7: $k = k + 1$.
8: **end while**

The Composite (functions) UnderEstimate Sequence Algorithm (CUESA) presented in Algorithm 1 solves problem (1) when $h = 0$. The algorithm uses a fixed step size $\alpha_k = \frac{\mu}{L}$, and note that y_k is not explicitly used in CUESA ($y_k = x_k$ for all $k \geq 0$). Moreover, the lower bound $\phi_{k+1}(x)$ is not explicitly constructed; the algorithm simply computes and uses the minimizer of the lower bound ϕ_{k+1}.

Considering Step 5 in isolation shows that each iteration of CUESA is simply a proximal gradient descent step. However, CUESA is distinct from a standard proximal gradient method due to the inclusion of additional ingredients related to the lower bound $\phi_k(x)$, which guarantee an -optimal solution.

In addition to the proximal gradient, the algorithm utilizes two vectors at every iteration, x_k and v_k. By (26), no additional vectors are required to update ϕ_k, but the function value $F(x_k^+) \equiv F(x_{k+1})$ is required. Overall, at every iteration of CUESA, three vectors must be stored, and one function value must be computed.

Theorem 1 *Let Assumption 1 hold. The sequences $\{x_k\}_{k=0}^{\infty}$, $\{\phi_k(x)\}_{k=0}^{\infty}$ and $\{\alpha_k\}_{k=0}^{\infty}$ generated by CUESA (Algorithm 1) form a UES.*

Proof It must be shown that the iterates generated by Algorithm 1 satisfy the conditions of Definition 1. Because $\alpha_k = \mu/L \quad (0, 1)$ for all $k \geq 0$, by Lemma 6, (4) holds. It remains to prove (5). From Step 4 in CUESA, one

sees that $y_k = x_k$ for all k, so it also follows that $y_k^+ = x_{k+1}$ for all k. Using $y = x = x_k$ in the lower bound (13) gives

$$F(x_{k+1}) \leq F(x_k) - \tfrac{1}{2L} \|G_L(x_k)\|^2. \tag{39}$$

Thus,

$$\begin{aligned}
&F(x_{k+1}) - \phi_{k+1} \\
&= (1-\alpha_k)F(x_{k+1}) + \alpha_k F(x_{k+1}) - \phi_{k+1} \\
&\overset{(29)}{=} (1-\alpha_k)F(x_{k+1}) + \alpha_k F(x_{k+1}) - (1-\alpha_k)\left[\phi_k + \alpha_k \tfrac{\mu}{2}\|v_k - y_k^+\|^2\right] \\
&\quad - \alpha_k \left[F(y_k^+) + \tfrac{1}{2L} - \tfrac{1}{2\mu}\|G_L(y_k)\|^2\right] \\
&= (1-\alpha_k)F(x_{k+1}) - (1-\alpha_k)\left[\phi_k + \alpha_k \tfrac{\mu}{2}\|v_k - y_k^+\|^2\right] \\
&\quad - \alpha_k \left[\tfrac{1}{2L} - \tfrac{1}{2\mu}\|G_L(y_k)\|^2\right] \\
&\leq (1-\alpha_k)\left(F(x_{k+1}) - \phi_k\right) - \alpha_k \left[\tfrac{1}{2L} - \tfrac{1}{2\mu}\right]\|G_L(y_k)\|^2 \\
&\overset{(39)}{\leq} (1-\alpha_k)\left[F(x_k) - \tfrac{1}{2L}\|G_L(x_k)\|^2 - \varphi_k\right] - \alpha_k \left[\tfrac{1}{2L} - \tfrac{1}{2\mu}\right]\|G_L(x_k)\|^2 \\
&\leq (1-\alpha_k)\left(F(x_k) - \varphi_k\right) + \left[\tfrac{\alpha_k}{2\mu} - \tfrac{\alpha_k}{2L} - (1-\alpha_k)\tfrac{1}{2L}\right]\|G_L(x_k)\|^2 \\
&\leq (1-\alpha_k)(F(x_k) - \phi_k),
\end{aligned}$$

where the last step follows because $\alpha_k = \frac{\mu}{L}$ so $\frac{\alpha_k}{2\mu} - \frac{1}{2L} = 0$.

Theorem 1 confirms that the iterates generated by Algorithm 1 form a UES. Recalling Proposition 1, and noting that $\sum_{k=0}^{\infty} \alpha_k = \infty$ because $\alpha_k = \mu/L$ for all $k \geq 0$, CUESA is guaranteed to converge linearly to the solution of (1), as is stated in the following corollary.

Corollary 1 *Let Assumption 1 hold. Then, the sequence of iterates $\{x_k\}_{k\geq 0}$ generated by Algorithm 1 exhibits a linear rate of convergence*

$$F(x_k) - \phi_k \leq \left(1 - \tfrac{\mu}{L}\right)^k (F(x_0) - \phi_0).$$

4.2 An Accelerated Composite UES Algorithm

CUESA has a linear rate of convergence, but the rate constant is suboptimal. Thus, the Accelerated Composite UnderEstimate Sequence Algorithm (ACUESA) for solving (1) when $h = 0$ is presented in Algorithm 2, and ACUESA achieves the best possible rate of convergence.

ACUESA uses a fixed step size $\alpha_k = \sqrt{\mu/L}$ and a fixed acceleration parameter $\beta_k = \frac{1}{1+\alpha_k}$. ACUESA explicitly uses a point $y_k (= x_k)$, which is a convex combination of the points x_k and v_k. At each iteration, a proximal gradient descent step is taken *from* y_k, with the proximal gradient computed at y_k. In addition to the proximal gradient, ACUESA constructs *three* points

Algorithm 2 Accelerated Composite UES Algorithm (ACUESA)

1: Initialization: Set $k = 0$, $\quad > 0$, initial point $x_0 \quad \mathrm{R}^n$ and compute μ, L.
2: Set v_0 and ϕ_0 as in (22). Let $\alpha_k = \quad \frac{\mu}{L}$, $\beta_k = \frac{1}{1+\alpha_k}$.
3: **while** $F(x_k) - \phi_k > \quad$ **do**
4: Set $y_k = \beta_k x_k + (1 - \beta_k) v_k$.
5: Set $x_{k+1} = y_k - \frac{1}{L} G_L(y_k)$.
6: Update v_{k+1} and ϕ_{k+1} as in (25) and (26) respectively.
7: $k = k + 1$.
8: **end while**

at every iteration, x_k, v_k and y_k. Thus, at every iteration of ACUESA, four n-dimensional vectors must be stored, and one function value must be computed.

Theorem 2 *Let Assumption 1 hold. The sequences $\{x_k\}_{k=0}^{\infty}$, $\{\phi_k(x)\}_{k=0}^{\infty}$ and $\{\alpha_k\}_{k=0}^{\infty}$ generated by ACUESA (Algorithm 2) form a UES.*

Proof From Step 5 in ACUESA,

$$x_{k+1} = y_k - \tfrac{1}{L} G_L(y_k) \overset{(9)}{\equiv} y_k^+. \tag{40}$$

Hence,

$$\begin{aligned} F(x_{k+1}) - \phi_{k+1} &= (1-\alpha_k)F(x_{k+1}) + \alpha_k F(x_{k+1}) - \phi_{k+1} \\ &\overset{(29)}{=} (1-\alpha_k)F(x_{k+1}) + \alpha_k F(x_{k+1}) - (1-\alpha_k)\phi_k - \alpha_k F(y_k^+) \\ &\quad -\alpha_k(1-\alpha_k)\tfrac{\mu}{2} \ v_k - y_k^+ \ ^2 - \alpha_k \ \tfrac{1}{2L} - \tfrac{1}{2\mu} \quad G_L(y_k) \ ^2 \\ &\overset{(40)}{=} (1-\alpha_k)F(x_{k+1}) - (1-\alpha_k)\phi_k \\ &\quad -\alpha_k(1-\alpha_k)\tfrac{\mu}{2} \ v_k - y_k^+ \ ^2 + \ \tfrac{\alpha_k}{2\mu} - \tfrac{\alpha_k}{2L} \quad G_L(y_k) \ ^2 \\ &= (1-\alpha_k) \ F(x_{k+1}) - \phi_k - \alpha_k \tfrac{\mu}{2} \ v_k - y_k^+ \ ^2 \\ &\quad + \ \tfrac{\alpha_k}{2\mu} - \tfrac{\alpha_k}{2L} \quad G_L(y_k) \ ^2. \end{aligned} \tag{41}$$

Rearranging the update for y_k in Step 4 of Algorithm 2 gives

$$v_k = \tfrac{1}{1-\beta_k}(y_k - \beta_k x_k) = y_k + \tfrac{\beta_k}{1-\beta_k}(y_k - x_k), \tag{42}$$

and because $\beta_k = \frac{1}{1+\alpha_k}$,

$$1 - \beta_k = 1 - \tfrac{1}{1+\alpha_k} = \tfrac{\alpha_k}{1+\alpha_k} = \alpha_k\beta_k \qquad \tfrac{\alpha_k\beta_k}{1-\beta_k} = 1. \tag{43}$$

Thus, combining (42) and (10) gives

$$\begin{aligned} &\tfrac{\alpha_k\mu}{2} \ v_k - y_k^+ \ ^2 \\ &= \tfrac{\alpha_k\mu}{2} \ \tfrac{\beta_k}{1-\beta_k}(y_k - x_k) + \tfrac{1}{\mu} G_L(y_k) \ ^2 \\ &= \tfrac{\alpha_k\mu}{2}\tfrac{\beta_k^2}{(1-\beta_k)^2} \ x_k - y_k \ ^2 + \tfrac{\alpha_k}{2\mu} \ G_L(y_k) \ ^2 - \tfrac{\alpha_k\beta_k}{1-\beta_k} \ G_L(y_k), x_k - y_k \\ &\overset{(43)}{=} \tfrac{\mu}{2}\tfrac{\beta_k}{1-\beta_k} \ x_k - y_k \ ^2 + \tfrac{\alpha_k}{2\mu} \ G_L(y_k) \ ^2 - \ G_L(y_k), x_k - y_k \ . \end{aligned} \tag{44}$$

Substituting (44) into (41) results in

$$\begin{aligned}
&F(x_{k+1}) - \phi_{k+1} \\
&= (1-\alpha_k)\left[F(x_{k+1}) - \phi_k - \tfrac{\mu}{2}\tfrac{\beta_k}{1-\beta_k}\|x_k - y_k\|^2 + \langle G_L(y_k), x_k - y_k\rangle\right] \\
&\quad -(1-\alpha_k)\tfrac{\alpha_k}{2\mu}\|G_L(y_k)\|^2 + \left[\tfrac{\alpha_k}{2\mu} - \tfrac{\alpha_k}{2L}\right]\|G_L(y_k)\|^2 \\
&= (1-\alpha_k)\left[F(x_{k+1}) - \phi_k - \tfrac{\mu}{2}\tfrac{\beta_k}{1-\beta_k}\|x_k - y_k\|^2 + \langle G_L(y_k), x_k - y_k\rangle\right] \\
&\quad +(1-\alpha_k)\tfrac{1}{2L}\|G_L(y_k)\|^2.
\end{aligned}$$

The second equality uses the fact that $\alpha_k = \sqrt{\mu/L}$. Using a rearrangement of the lower bound (13), and (40), gives

$$\begin{aligned}
&F(x_{k+1}) - \phi_{k+1} \\
&\le (1-\alpha_k)\left[F(x_k) - \langle G_L(y_k), x_k - y_k\rangle - \tfrac{\mu}{2}\|x_k - y_k\|^2 - \tfrac{1}{2L}\|G_L(y_k)\|^2 - \phi_k\right] \\
&\quad +(1-\alpha_k)\left[-\tfrac{\mu}{2}\tfrac{\beta_k}{1-\beta_k}\|x_k - y_k\|^2 + \langle G_L(y_k), x_k - y_k\rangle + \tfrac{1}{2L}\|G_L(y_k)\|^2\right] \\
&= (1-\alpha_k)(F(x_k) - \phi_k) + (1-\alpha_k)\left[-\tfrac{\mu}{2}\|x_k - y_k\|^2 - \tfrac{\mu}{2}\tfrac{\beta_k}{1-\beta_k}\|x_k - y_k\|^2\right] \\
&\le (1-\alpha_k)(F(x_k) - \phi_k) - \tfrac{\mu}{2}(1-\alpha_k)\tfrac{1}{1-\beta_k}\|x_k - y_k\|^2 \\
&\le (1-\alpha_k)(F(x_k) - \phi_k).
\end{aligned}$$

Thus, the iterates generated by ACUESA form a UES.

Theorem 2 confirms that the iterates generated by Algorithm 2 form a UES, and because $\alpha_k = \sqrt{\mu/L}$, ACUESA is guaranteed to converge linearly at the optimal rate to the solution of problem (1), as is shown in the following corollary.

Corollary 2 *Let Assumption 1 hold. Then, the sequence of iterates $\{x_k\}_{k\ge 0}$ generated by Algorithm 2 exhibits the optimal linear rate of convergence*

$$F(x_k) - \phi_k \le \left(1 - \sqrt{\tfrac{\mu}{L}}\right)^k (F(x_0) - \phi_0).$$

The difference in convergence rates between Algorithms 1 and 2 is essentially explained by the quadratic term $\|v_k - y_k^{++}\|^2$, which is entirely ignored in the proof of Theorem 1. In the proof of Theorem 2, one is able to incorporate an additional term containing $\|G_L(y_k)\|^2$, which leads to a larger allowable value of α_k, and ultimately, a tighter bound for Algorithm 2.

5 Algorithms and Convergence Guarantees for Smooth Functions

Here smooth optimization problems (problem (1) with $h \equiv 0$) are considered, and two gradient descent type algorithms whose iterates form a UES are presented. The algorithms are similar to those for composite problems, but they use the lower bounds developed in Section 3.3.

5.1 UES Algorithms for Smooth Functions

Algorithm 3 Smooth Underestimate Sequence Algorithm (SUESA)

1: Initialization: Set $k = 0$, $\quad > 0$, initial point $x_0 \quad \mathrm{R}^n$ and compute μ, L.
2: Set v_0 and φ_0 as in (35), and let $\alpha_k = \frac{\mu}{L}$.
3: **while** $f(x_k) - \varphi_k > \quad$ **do**
4: Set $y_k = x_k$, $y_k^+ = x_k^+$ and $y_k^{++} = x_k^{++}$.
5: Set $x_{k+1} = x_k - \frac{1}{L} \quad f(x_k)$.
6: Update v_{k+1} and φ_{k+1} as in (25) and (38), respectively.
7: $k = k + 1$.
8: **end while**

Algorithm 4 Accelerated Smooth Underestimate Sequence Algorithm (ASUESA)

1: Initialization: Set $k = 0$, $\quad > 0$, initial point $x_0 \quad \mathrm{R}^n$ and compute μ, L.
2: Set v_0 and φ_0 as in (35). Let $\alpha_k = \quad \frac{\mu}{L}$ and $\beta_k = \frac{1}{1+\alpha_k}$.
3: **while** $f(x_k) - \varphi_k > \quad$ **do**
4: Set $y_k = \beta_k x_k + (1 - \beta_k) v_k$.
5: Set $x_{k+1} = y_k - \frac{1}{L} \quad f(y_k)$.
6: Update v_{k+1} and φ_{k+1} as in (25) and (38), respectively.
7: $k = k + 1$.
8: **end while**

SUESA (Algorithm 3) is similar to CUESA, and ASUESA (Algorithm 4) is similar to ACUESA. In the smooth case the proximal gradient is simply the gradient so that the main update in SUESA (see Step 5) is a gradient descent step. The minimizer of the lower bound φ_{k+1} is updated in place of ϕ_{k+1} — this variation means that SUESA is *not* an instance of CUESA when $h = 0$, nor is ASUESA and instance of ACUESA (see Section 5.3 for further details).

Theorem 3 *Let Assumption 1 hold. The sequences* $\{x_k\}_{k=0}^{\infty}$, $\{\varphi_k(x)\}_{k=0}^{\infty}$ *and* $\{\alpha_k\}_{k=0}^{\infty}$ *generated by SUESA (Algorithm 3) form a UES.*

Proof By Lemma 8, (4) holds. To prove (5), combining the definition of x_{k+1} (Step 5 in Algorithm 3) with (3), and noting that $x_k = y_k$, gives

$$f(x_{k+1}) = f(y_k^+) \leq f(y_k) - \frac{1}{2L} \quad f(y_k)^{\,2}. \tag{45}$$

It is straightforward to establish that $f(x_{k+1}) - \varphi_{k+1} \leq (1 - \alpha_k)(f(x_k) - \varphi_k)$ using (38) and (45), and noting that $\alpha_k = \mu/L \quad (0, 1)$ for all $k \geq 0$.

Corollary 3 *Let Assumption 1 hold. Then the sequences* $\{x_k\}_{k=0}^{\infty}$, $\{\varphi_k(x)\}_{k=0}^{\infty}$ *and* $\{\alpha_k\}_{k=0}^{\infty}$ *generated by SUESA (Algorithm 3) form a UES, so SUESA converges at a linear rate:* $f(x_k) - \varphi_k \leq (1 - \frac{\mu}{L})^k (f(x_0) - \varphi_0)$.

Theorem 4 *Let Assumption 1 hold. The series of sequences $\{x_k\}_{k=0}^{\infty}$, $\{\varphi_k(x)\}_{k=0}^{\infty}$ and $\{\alpha_k\}_{k=0}^{\infty}$ generated by ASUESA in Algorithm 4 form a UES.*

Proof Note that, rearranging the expression for y_k in Step 4 of Algorithm 4 gives (42) and because $\beta_k = \frac{1}{1+\alpha_k}$, (43) holds. Thus, by the convexity of f we have

$$\begin{aligned} -\alpha_k \quad f(y_k), v_k - y_k &\overset{(42)}{=} -\alpha_k \quad f(y_k), y_k + \tfrac{\beta_k}{1-\beta_k}(y_k - x_k) - y_k \\ &\leq \tfrac{\alpha_k\beta_k}{1-\beta_k}\left[f(x_k) - f(y_k)\right] \overset{(43)}{=} f(x_k) - f(y_k). \quad (46) \end{aligned}$$

To confirm that $f(x_{k+1}) - \varphi_{k+1} \leq (1-\alpha_k)(f(x_k) - \varphi_k)$, combine (38) and (45), complete the square, apply (46) and note that $\alpha_k = \frac{\mu}{L}$.

Corollary 4 *Let Assumption 1 hold. Then the sequences $\{x_k\}_{k=0}^{\infty}$, $\{\varphi_k(x)\}_{k=0}^{\infty}$ and $\{\alpha_k\}_{k=0}^{\infty}$ generated by ASUESA (Algorithm 4) form a UES, so ASUESA converges at the optimal linear rate: $f(x_k) - \varphi_k \leq (1 - \frac{\mu}{L})^k (f(x_0) - \varphi_0)$.*

5.2 A Toy Example

Notice that for SUESA, at iteration $k \geq 0$, the distance between x_k and the minimizer of the lower bound shrinks at a fixed rate. That is, since $\alpha_k = \frac{\mu}{L}$, the following equality holds:

$$\begin{aligned} x_{k+1} - v_{k+1} &= \left[x_k - \tfrac{1}{L} \quad f(x_k)\right] - \left[(1-\alpha_k)v_k + \alpha_k(x_k - \tfrac{1}{\mu} \quad f(x_k))\right] \\ &= (1 - \tfrac{\mu}{L})(x_k - v_k). \quad (47) \end{aligned}$$

Equation (47) illustrates that, after each iteration of Algorithm 3, the line joining x_{k+1} and v_{k+1} is parallel to the line joining x_k and v_k. Moreover, the distance between the two points is reduced by precisely $(1 - \frac{\mu}{L})$ at every iteration. Intuitively, x_k and the minimizer v_k are becoming ever closer, and eventually they both converge to x (recall Theorem 3).

This fact can be visualized via the following toy example. Consider the (smooth) regularized logistic regression problem, i.e., problem (1) with $h = 0$ and

$$f(x) = \sum_{i=1}^{m} \log(1 + \exp(-y_i \langle x, a_i \rangle)) + \tfrac{\lambda}{2} \|x\|^2,$$

where $a_i \in \mathbb{R}^n$ is the ith feature vector with corresponding (binary) label $y_i \in \mathbb{R}$. For this example, 100 two dimensional data points with binary labels $\{a_i, y_i\}$ were randomly generated (so $m = 100$ and $n = 2$), and these points and labels are shown in the top left plot in Figure 2. (Each point a_i is plotted on a 2D grid, and the point is colored green or red to highlight its label y_i). Parameter $\lambda = 0.01$, so the strong convexity constant is $\mu = 0.01$. SUESA and ASUESA were used to solve this problem, starting from the

point $x_0 = (-20, 10)^T$. Both algorithms were stopped after 100 iterations and the results of this experiment are shown in Figure 2. The optimal solution is $x^* = (-0.2610, 0.2407)^T$ with corresponding optimal value $F^* \equiv f^* = 0.6779$ (all to 4dp).

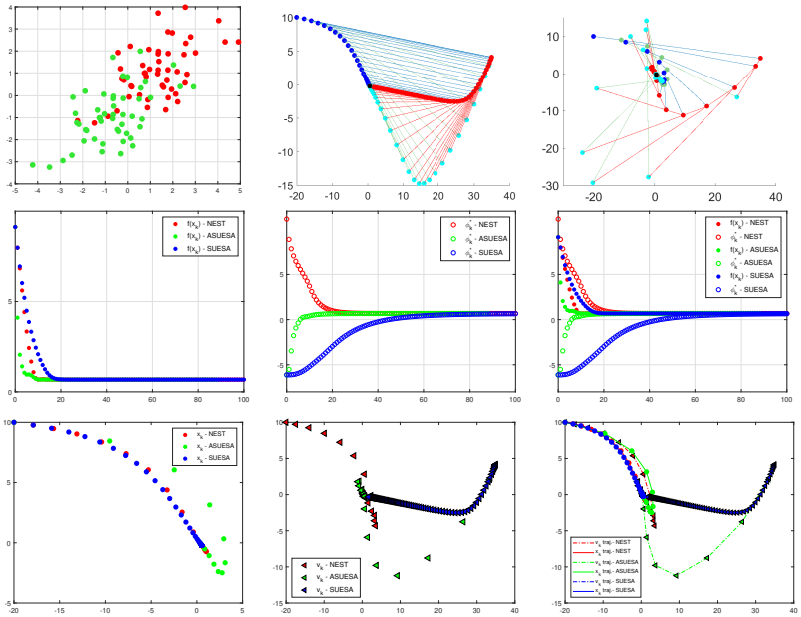

Fig. 2 SUESA and ASUESA applied to the toy example described in Section 5.2. Top row from left: the simulated data; the iterates generated by SUESA; the iterates generated by ASUESA. Middle row from left: the function values $f(x_k)$ for SUESA, ASUESA and NEST; the values φ_k for SUESA, ASUESA and NEST algorithm; both $f(x_k)$ and φ_k for SUESA, ASUESA and NEST. Bottom row from left: the iterates x_k generated by SUESA, ASUESA and NEST; the points v_k generated by SUESA, ASUESA and NEST; both x_k and v_k for SUESA, ASUESA and NEST.

Consider the iterates generated by SUESA in Figure 2. The blue dots represent the iterates x_k, starting from $x_0 = (-20, 10)^T$. The aqua dots are the long steps x_k^{++}. Notice that the iterates (short steps) x_k, $x_k^+ (\equiv x_{k+1})$ and long steps x_k^{++} all lie on the same (aqua) line. This is simply because x_k^+ and x_k^{++} are the result of starting from x_k and then moving in the direction $-\nabla f(x_k)$, scaled by $1/L$ or $1/\mu$, respectively. So all the aqua lines correspond to the negative gradient directions. The red dots correspond to the points v_k. Initially $v_0 = x_0^{++} \approx (35, 5)^T$ so the first red and aqua point coincide. The blue lines join x_k and v_k, and they are all parallel. As SUESA progresses, the points x_k (blue) and v_k (red) eventually converge to the minimizer (black dot).

The last plot on the first row in Figure 2 shows the iterates of ASUESA. Additionally for ASUESA, the green points correspond to the iterates y_k, and one notices that these points lie on the line joining x_k and v_k.

The second row of Figure 2 shows (left) the function values $f(x_k)$, (middle) the bounds φ_k, and (right) both the function values $f(x_k)$ and bounds φ_k for SUESA, ASUESA and NEST for the toy problem. The middle plot shows that the bounds φ_k for SUESA and ASUESA are *lower bounds for* $f(x_k)$, while the bound φ_k for NEST is an upper bound for $f(x_k)$. This is why a verifiable stopping condition can be used for SUESA and ASUESA (the function values decrease, the lower bounds increase, and the gap sandwiches the optimal function value $f(x\ )$, squeezing closer to it). On the other hand, for NEST, φ_k pushes $f(x_k)$ down from above, but provides no information about how close the current value $f(x_k)$ is from the optimal value $f(x\ )$.

The third row of Figure 2 shows (left) the iterates x_k, (middle) the points v_k, and (right) both the iterates x_k and the points v_k for SUESA, ASUESA and NEST for the toy problem. The left and middle plots clearly show that the iterates x_k and points v_k are di□erent for each algorithm. *ASUESA is not the same as NEST*. The trajectory of the iterates x_k (and the points v_k) are distinct for each algorithm. This confirms that the algorithms presented here are new, and are not simply an existing algorithm with an added lower bound.

5.3 A comparison of the composite and smooth algorithms when $h \equiv 0$

A natural question to ask is, 'Do the composite algorithms (CUESA and ACUESA) recover the smooth algorithms (SUESA and ASUESA) when $h \equiv 0$?'. Ultimately, the answer to this question is no, but this is a consequence of the termination condition (i.e., the lower bounds), rather than the path of the iterates.

Table 1 compares the vectors generated by ASUESA (Algorithm 4) and ACUESA (Algorithm 2) in the smooth case for a given initial point $x_0 \quad \mathrm{R}^n$. Note that the same values α_k and β_k are used by both algorithms, and for $h \equiv 0$, $G_L(y_k) \equiv \quad f(y_k)$. Table 1 confirms that the iterates generated by ASUESA and ACUESA follow the same path.

Variable	ACUESA($h \equiv 0$)		ASUESA	
v_0	x_0^{++}	(22)	x_0^{++}	(35)
y_k	$\beta_k x_k + (1-\beta_k)v_k$	Step 4	$\beta_k x_k + (1-\beta_k)v_k$	Step 4
x_{k+1}	$y_k - \frac{1}{L} \quad f(y_k)$	Step 5	$y_k - \frac{1}{L} \quad f(y_k)$	Step 5
v_{k+1}	$(1-\alpha_k)v_k + \alpha_k y_k^{++}$	(25)	$(1-\alpha_k)v_k + \alpha_k y_k^{++}$	(25)

Table 1 The vectors generated by ACUESA and ASUESA when $h \equiv 0$.

However, consider the stopping gap for each algorithm, i.e., compare

$$f(x_{k+1}) - \varphi_{k+1} \overset{(38)}{=} f(x_{k+1}) - (1-\alpha_k)\alpha_k \tfrac{\mu}{2} \|v_k - y_k^+\|^2 + \tfrac{\alpha_k}{2\mu} \|f(y_k)\|^2 - (1-\alpha_k)\varphi_k - \alpha_k f(y_k). \quad (48)$$

with

$$f(x_{k+1}) - \phi_{k+1} \overset{(29)}{=} f(x_{k+1}) - (1-\alpha_k)\alpha_k \tfrac{\mu}{2} \|v_k - y_k^+\|^2 + \tfrac{\alpha_k}{2\mu} \|f(y_k)\|^2 - (1-\alpha_k)\phi_k - \alpha_k f(y_k^+) - \alpha_k \tfrac{1}{2L} \|f(y_k)\|^2. \quad (49)$$

At every iteration, by (45) it holds that $-f(y_k) \leq -f(y_k^+) - \frac{\alpha_k}{2L} \|f(y_k)\|^2$ and so we conclude that $f(x_{k+1}) - \varphi_{k+1} \leq f(x_{k+1}) - \phi_{k+1}$ for all $k \geq 0$. This confirms that, in the smooth case with $h \equiv 0$, ACUESA does not recover ASUESA. Moreover, it is advantageous to use ASUESA because the algorithm will terminate sooner than ACUESA for $h \equiv 0$.

Similar arguments can be used to show that CUESA and SUESA are not the same algorithm in the smooth case, and that it is advantageous to use SUESA because SUESA will terminate earlier than CUESA.

6 An algorithm with adaptive L

In all of the algorithms presented so far, the Lipschitz constant L is explicitly used. However, by studying the convergence proofs for Algorithms 1–4 one observes that the role of L is to enforce a reduction in the function value from one iteration to the next. Thus, it is natural to ask the question, 'Can an *adaptive* Lipschitz constant, say L_k, be used in place of the true Lipschitz constant L, while preserving convergence guarantees?'. This is discussed now.

When the Lipschitz constant L is unknown, or is expensive to compute, it may be preferable to employ an 'adaptive' Lipschitz constant, say L_k, that approximates L locally. This approach, studied by Nesterov in [23,25], has the additional advantage that L_k may be smaller than the true Lipschitz constant L, which can lead to large step sizes. To maintain convergence properties for Algorithms 1–4, certain inequalities must hold; the relevant inequalities are as follows.

Non-smooth case. For composite functions, (39) and (41) must hold for CUESA and ACUESA, respectively. So, if L_k satisfies

$$F\left(y_k - \tfrac{1}{L_k} G_{L_k}(y_k)\right) \leq F(y_k) - \tfrac{1}{2L_k} \|G_{L_k}(y_k)\|^2, \quad (50)$$

then the algorithms are still guaranteed to converge. If L_k satisfies (50), then one obtains the improvement $\alpha_k = \mu/L_k$ (or $\alpha_k = \sqrt{\mu/L_k}$ for the accelerated case) at every iteration.

Smooth case. For smooth functions, (45) must hold for SUESA and ASUESA. This means that at every iteration, if L_k satisfies

$$f\left(y_k - \tfrac{1}{L_k}\nabla f(y_k)\right) \leq f(y_k) - \tfrac{1}{2L_k}\|\nabla f(y_k)\|^2, \tag{51}$$

then convergence guarantees for SUESA and ASUESA are maintained. If L_k satisfies (51) then we have the improvement $\alpha_k = \mu/L_k$ (or $\alpha_k = \sqrt{\mu/L_k}$ for the accelerated case) at every iteration.

With these two inequalities in mind, the adaptive Lipschitz process is described now. Initialize Algorithms 1–4 with the estimate $0 < \mu \leq L_0 \leq L$, and increase and decrease factors $u > 1$ and $d \geq 1$, respectively. At each iteration $k \geq 1$ of Algorithms 1–4, an inner loop (described in Algorithm 5) is employed to find the approximation L_k. The inner loop (Algorithm 5) replaces Steps 4–5 in Algorithms 1–4. The psuedocode is presented now, and the inner iteration counter s in used in Algorithm 5 to distinguish it from the outer loop iteration counter k.

Algorithm 5 Finding L_k in iteration k of Algorithms 4 and 2.

Input: x_k, v_k $u > 1$, $d \geq 1$, $0 < \mu < L_0 \leq L$ and L_{k-1}.
Initialize: $L_k^{(s)} = \max\{L_0, L_{k-1}/d\}$.
for $s = 0, 1, 2, \ldots$ **do**
  $\alpha_k^{(s)} = \sqrt{\frac{\mu}{L_k^{(s)}}}$, $\beta_k^{(s)} = \frac{1}{1+\alpha_k^{(s)}}$.
  Set $y_k^{(s)} = \beta_k^{(s)} x_k + (1 - \beta_k^{(s)}) v_k$.
  Set $x_{k+1}^{(s)} = y_k^{(s)} - \frac{1}{L_k^{(s)}} G_{L_k^{(s)}}(y_k^{(s)})$.
  if (51) or (50) holds **then**
    Break.
  else
    $L_k^{(s+1)} = u \cdot L_k^{(s)}$.
  end if
end for
Output: $L_k = L_k^{(s)}$, $\alpha_k = \alpha_k^{(s)}$, $\beta_k = \beta_k^{(s)}$, $y_k = y_k^{(s)}$ and $x_{k+1} = x_{k+1}^{(s)}$.

Algorithm 5 begins with the estimate $L_k^{(0)} = \max\{L_0, L_{k-1}/d\}$. This ensures that (i) $L_k \geq L_0 \geq \mu > 0$ so that convergence of the outer loop is maintained (because $\alpha_k = \mu/L_k \in (0, 1)$) must hold); and allows that possibility that (ii) $L_k^{(0)}$ is $\frac{1}{d}$ times smaller than L_{k-1} so that large step sizes might be used. Next, $L_k^{(s)}$ is (possibly repeatedly) multiplied by the increase factor u until (51) (or (50)) is satisfied at inner iteration S, at which point, $L_k = L_k^{(S)}$ is passed to the outer loop and iteration k continues with L_k used in place of L. Following this process, it possibly occurs that at some iteration k, $L_k < L$. In this case, the stepsize $1/L_k$ is used, which is larger than $1/L$.

The strategy above holds for Algorithms 2 and 4, but it is straightforward to adapt it to Algorithms 1 and 3 by modifying the variables as $\alpha_s = \mu/L_s$ and $\beta_s = 1$.

Using the arguments in [25, Section 3], the value L_k can increase only if $L_k \leq L$. Thus, by Assumption 1 (Lipschitz continuity), the following chain of inequalities holds

$$0 < \mu \leq L_0 \leq L_k \leq u \cdot L. \tag{52}$$

The following lemma bounds the number of inner iterations.

Lemma 9 (Lemma 4 in [25]) *Let Assumption 1 hold, and let $u > 1$, $d \geq 1$ and $0 < \mu \leq L_0 \leq L$. Then the maximum number of times that Lines 4 to 10 of Algorithm 5 are executed during the first K iterations of Algorithm 2 (or Algorithm 4), say N_k, is bounded by*

$$N_K \leq \left\lceil 1 + \frac{\ln d}{\ln u} \right\rceil \cdot (K+1) + \frac{1}{\ln u} \max \left\{ \ln \frac{u \cdot \ln L}{d \cdot \ln L_0}, 0 \right\}. \tag{53}$$

7 Numerical Experiments

In this section, we present numerical results to compare our proposed algorithms with several other methods that have an optimal convergence rate. The algorithms are as follows, and are summarized in Table 2.

OQA/OQA+. The Optimal Quadratic Averaging algorithm (OQA) [8], which builds upon the work in [3], maintains a quadratic lower bound on the objective function value at every iteration. The quadratic lower bound is called 'optimal' because it is the 'best' lower bound that can be obtained as a convex combination of the previous 2 quadratic lower bounds. In OQA, x_{k+1} is set to be the minimizer of $f(x)$ on the line joining the points x_k^+ and the minimizer of the current quadratic lower bound. In [8] the author suggest a variant of OQA, which we call OQA+ here, that computes x_k^+ via a line search that does not use the true Lipschitz constant L.

NEST/NEST+. NEST denotes the algorithm described in Chapter 2 of [21], which is for *smooth* functions. Further, NEST+ is a variant of NEST in which the Lipschitz constant L is adaptively update via the strategy in [23,25]. Both variants are *accelerated* algorithms.

CNEST/CNEST+. CNEST denotes Algorithm (4.9) in [23], which can be applied to *composite nonsmooth* functions. Further, CNEST+ is a variant of CNEST in which the Lipschitz constant L is adaptively update via the strategy in [23, 25]. Both variants are *accelerated* algorithms.

GD. We also implement a Gradient Descent (GD) method which uses a fixed stepsize of $\frac{1}{L}$. Note that this is similar to Algorithm 3, although GD does not maintain any kind of lower bound. As the only non-optimal algorithm, Gradient Descent provides a benchmark that will enable us to observe any performance advantages of the optimal methods.

Algorithm	Description
OQA	Optimal Quadratic Averaging Algorithm
OQA+	Optimal Quadratic Averaging Algorithm with $x_k^+ = \text{line-search}(x_k, x_k - f(x_k))$
SUESA	Smooth Underestimate Sequence Algorithm
ASUESA	Accelerated Smooth Underestimate Sequence Algorithm
ASUESA+	Accelerated Smooth Underestimate Sequence Algorithm with adaptive Lipschitz constant
NEST	Accelerated algorithm described in Chapter 2 of [21]
NEST+	Accelerated algorithm described in Chapter 2 of [21] with adaptive Lipschitz constant
CUESA	Composite Underestimate Sequence Algorithm
CUESA+	Composite Underestimate Sequence Algorithm with adaptive Lipschitz constant
ACUESA	Accelerated Composite Underestimate Sequence Algorithm
ACUESA+	Accelerated Composite Underestimate Sequence Algorithm with adaptive Lipschitz constant
CNEST	Accelerated Algorithm (4.9) in [23] with fixed Lipschitz constant
CNEST+	Accelerated Algorithm (4.9) in [23] with adaptive Lipschitz constant

Table 2 Description of implemented algorithms

7.1 Empirical Risk Minimization

We consider two Empirical Risk Minimization (ERM) problems, which are popular in the machine learning literature. In particular, we study ERM with a squared hinge loss

$$f(x) = \frac{1}{m}\sum_{i=1}^{m} \max\{0, 1 - y_i a_i^T x\}^2 + \frac{\lambda}{2}\|x\|^2, \tag{54}$$

and ERM with a logistic loss (also called logistic regression)

$$f(x) = \frac{1}{m}\sum_{i=1}^{m} \log\left(1 + e^{-y_i a_i^T x}\right) + \frac{\lambda}{2}\|x\|^2. \tag{55}$$

In each case $y_i \in \{-1, +1\}$ is the label and $a_i \in \mathbb{R}^n$ represents the training data for $i = 1, 2, \ldots, m$. All the datasets in our experiments come from LIBSVM database [4]. Also note that for all experiments we have $\mu = \lambda$.

Note that in these experiments, when implementing a line search in direction v (whenever relevant), the dot products $a_i^T v$ are reused.

Comparison on Decreasing Objective Values

In the first experiment we compare the OQA, ASUESA and NEST algorithms (both the standard and adaptive Lipschitz variants) and investigate how the objective function values behave on several test problems. The test problems considered in this experiment are the `a1a` dataset with a squared hinge loss and a value $\lambda = 10^{-4}$, the `rcv1` dataset with a logistic loss and a value $\lambda = 10^{-4}$, and the `covtype` dataset with a squared hinge loss and a value $\lambda = 10^{-5}$. In Figure 3 we plot the gap $f(x_k) - \varphi_k$ vs the number of function evaluations and the gap $f(x_k) - \varphi_k$ vs the cpu time. The figure shows the advantages of using an adaptive Lipschitz constant with the adaptive methods performing better than their original versions in most cases. Figure 3 also shows that ASUESA+ performs very well, being the best algorithm on the first dataset, and the second best algorithm on the other two datasets. The numerical experiments highlight that the gap, $f(x_k) - \varphi_k$, goes to zero *monotonically* for our

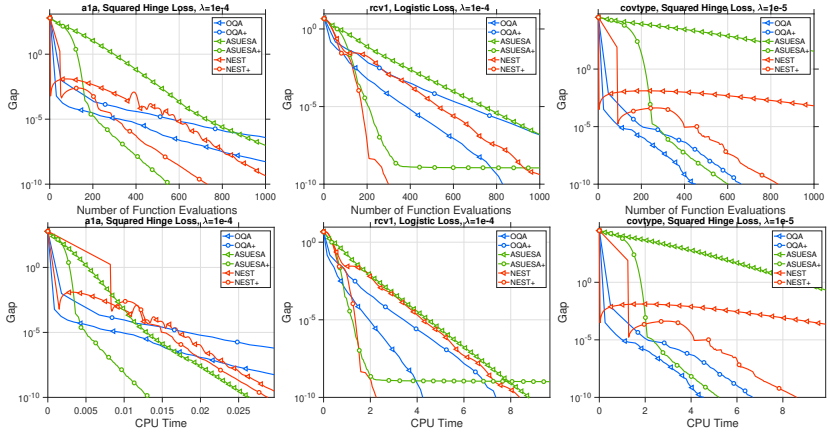

Fig. 3 Evolution of the gap $f(x_k) - \varphi_k$ for each algorithm compared with the number of function evaluations and cputime.

proposed methods, which matches our theoretical results. It is crucial to note that this does *not* necessarily mean that the objective function value decreases monotonically. Note also that the results confirm that Nesterov's method is non-monotone as well (which is also observed in the study [27]).

Theory and Practice for OQA and ASUESA

In this numerical experiment we study ASUESA and OQA and investigate how their practical performance compares with that predicted by theory. For the OQA algorithm a line search is needed to determine a necessary algorithmic variable, and to ensure that theory for OQA holds, the line search should be exact. In this experiment we will use bisection to compute this variable, but we will restrict the number of bisection steps allowed to $b = 2, 5, 20$. Figure 4 plots the ratio $(f(x_k) - \varphi_k)/(f(x_{k-1}) - \varphi_{k-1})$ for ASUESA and for three instances of OQA, where each instance uses a different number of bisection steps $b = 2, 5, 20$. We also plot $1 - \sqrt{\frac{\mu}{L}}$ (black dots), which is the amount of decrease in the gap $f(x_k) - \varphi_k$ at each iteration predicted by the theory. (In theory, we should have $(f(x_k) - \varphi_k)/(f(x_{k-1}) - \varphi_{k-1}) \leq 1 - \sqrt{\frac{\mu}{L}}$ for all $k \geq 0$.)

From the plots in Figure 4 we see that ASUESA performs very well, and as predicted by the theory, with the ratio $(f(x_k) - \varphi_k)/(f(x_{k-1}) - \varphi_{k-1})$ always strictly below the theoretical bound. On the other hand, the quality of line search affects OQA significantly. The fewer the number of line search (bisection) iterations, the more likely it is for OQA to violate the theoretical results. Note that this is not necessarily surprising because the theory for OQA requires the exact minimizer along a line segment to be found, so 2 or 5 iterations of bisection may be simply too few to find it. Notice that when $b = 2$, the green line shows that OQA behaves erratically, with the ratio $(f(x_k) - \varphi_k)/(f(x_{k-1}) - \varphi_{k-1})$ being greater than 1 on many iterations,

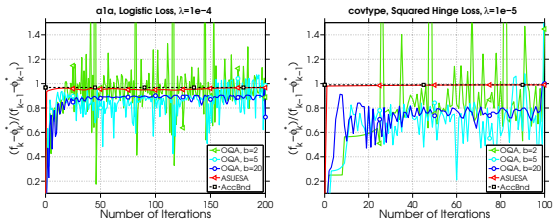

Fig. 4 Comparison of $\frac{f(x_k)-\varphi_k}{f(x_{k-1})-\varphi_{k-1}}$ for ASUESA and for OQA with different numbers of bisection steps ($b = 2, 5, 20$). The black dots are $1 - \sqrt{\frac{\mu}{L}}$.

indicating that the gap is growing on those iterations. When we use OQA with $b = 5$ steps of bisection at each iteration (light blue line), the algorithm performs better, and often, but not always, the ratio is less than 1. Finally, the dark blue line shows the behaviour of OQA when $b = 20$ steps of bisection at each iteration. The dark blue line is always below the theoretical bound of $1 - \sqrt{\frac{\mu}{L}}$, indicating good algorithmic performance (often better than predicted by theory). However, the line search needed by OQA comes at an additional computational cost, which can still mean that the overall runtime is longer for OQA than for ASUESA, as we now show.

Here a similar experiment is performed to compare the theoretical and practical performance of SUESA and ASUESA. We have already seen that the theoretical results for ASUESA give a proportional reduction of $1 - \sqrt{\frac{\mu}{L}}$ in the gap at every iteration. However, for SUESA, the proportional reduction in the gap is $1 - \frac{\mu}{L}$. We investigate how these theoretical bounds compare with the practical performance of each of these algorithms. We use the `a1a`, `rcv1` and `covtype` datasets for this experiment, and for each of the three datasets we form both a logistic loss, and a squared hinge loss to create 6 problem instances. The results are shown in Figure 5. Figure 5 presents the ratio $\frac{f(x_k)-\varphi_k}{f(x_{k-1})-\varphi_{k-1}}$ for SUESA and ASUESA. Also displayed is the theoretical (unaccelerated) rate $1 - \frac{\mu}{L}$ (the green line) and the theoretical (accelerated) rate $1 - \sqrt{\frac{\mu}{L}}$ (the black line). One sees that the practical performance of SUESA is very similar to that predicted by the theory because the blue line matches the green line closely. Another observation is that for the accelerated algorithm (ASUESA), in practice, the reduction in the gap $f(x_k) - \varphi_k$ is often more optimistic than the theoretical rate.

7.2 Experiments on composite functions

In this section we perform several numerical experiments on problems with a composite objective. Specifically, we consider the elastic net problem, which is

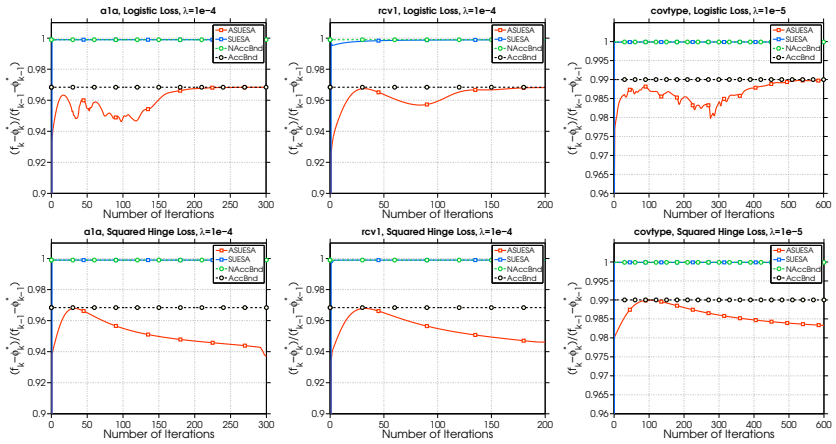

Fig. 5 Comparison of $\frac{f(x_k)-\varphi_k}{f(x_{k-1})-\varphi_{k-1}}$ for SUESA and ASUESA and $1-\frac{\mu}{L}$ (green line) and $1-\sqrt{\frac{\mu}{L}}$ (black line).

problem (1) with

$$F(x) = \frac{1}{n}\sum_{i=1}^{n}(a_i^T x - y_i)^2 + \frac{\lambda_1}{2}\|x\|^2 + \frac{\lambda_2}{2}\|x\|_1. \tag{56}$$

Notice that the first two terms in (56) are smooth, while the $_1$-norm term makes (56) nonsmooth overall. We compare our Algorithm 1 and 2 (CUESA and ACUESA) with the one proposed in [23] (CNEST). As stated previously, each of these algorithms can be implemented with either a fixed L or an adaptive L, and we will compare each algorithm under both of these two options.

For these experiments we again use the 3 datasets `a1a`, `rcv1` and `covtype`. For the `a1a` data the regularization parameters were set to $\lambda_1 = \lambda_2 = 10^{-4}$, for the `rcv1` data the regularization parameters were set to $\lambda_1 = 10^{-4}$ and $\lambda_2 = 10^{-5}$, and for the `covtype` data the regularization parameters were set to $\lambda_1 = 10^{-4}$ and $\lambda_2 = 10^{-6}$. The results of this experiment are presented in Figure 6, and they show the clear practical advantage of the ACUESA algorithm. The ACUESA algorithm outperforms the CNEST algorithm in all problem instances. Interestingly, on the `rcv1` dataset, the CUESA+ algorithm (CUESA with an adaptive Lipschitz constant) performs better than the accelerated ACUESA algorithm, although the ACUESA+ (accelerated plus adaptive Lipschitz constant) algorithm is still the best overall.

In the final numerical experiment presented here, we investigate the theoretical vs practical performance of CUESA and ACUESA. We set up three problems using each of the 3 datasets already described, and the results are presented in Figure 7.

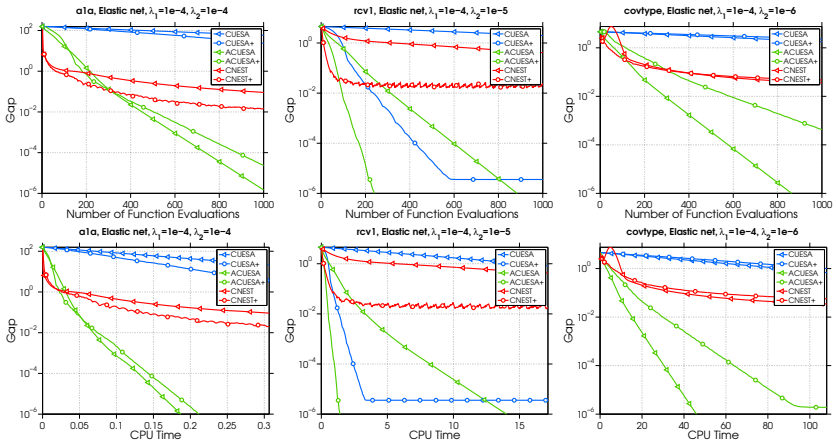

Fig. 6 Comparison of how gaps between the objective values and the minimum amount of the lower bounds decreases for different algorithms. We observe the advantage of ACUESA+ in both number of function evaluations and running time.

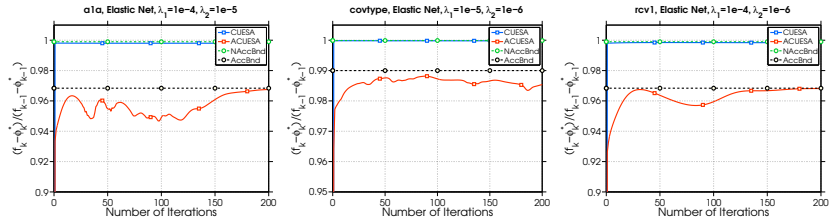

Fig. 7 Comparison of $\frac{f(x_k)-\varphi_k}{f(x_{k-1})-\varphi_{k-1}}$ for CUESA and ACUESA and $1-\frac{\mu}{L}$ (green line) and $1-\sqrt{\frac{\mu}{L}}$ (black line). Here we have a similar observation as in Figure 6.

As before, the green line represents the theoretical (unaccelerated) rate $1-\frac{\mu}{L}$ and the black line represents the theoretical (accelerated) rate $1-\sqrt{\frac{\mu}{L}}$. Note that the practical performance of CUESA closely matches the theoretical rate. We also observe that the practical performance of ACUESA is always at least as good as the theoretical rate, and can often get better decrease in the gap per iteration than $1-\sqrt{\frac{\mu}{L}}$.

All the numerical results presented in this section strongly support the practical success of the SUESA, ASUESA, CUESA and ACUESA algorithms.

8 Conclusion

In this paper we studied efficient algorithms for solving the strongly convex composite problem (1). Four new algorithms were proposed — CUESA, ACUESA, SUESA and ASUESA — to solve (1) in both the composite and

smooth cases. Each algorithm maintains a global lower bound on the objective function value, which can be used as an algorithm stopping condition to ensure and −optimal solution. Moreover, the definition of a new underestimate sequence that incorporates three sequences, one of which is a global lower bound on the objective function, and this framework was used to establish convergence guarantees for the algorithms proposed here. Our algorithms have a linear rate of convergence, and the two accelerated variants (ACUESA and ASUESA) converge at the optimal linear rate. We also presented a strategy to adaptively select a local Lipschitz constant for the situation when one does not wish to, or cannot, compute the true Lipschitz constant. Numerical experiments show that our algorithms are computationally competitive when compared with other state-of-the-art methods including Nesterov's accelerated gradient methods and optimal quadratic averaging methods.

Acknowledgement

This work was partially supported by the U.S. National Science Foundation, under award numbers NSF:CCF:1618717 and NSF:CCF:1740796.

References

1. Zeyuan Allen-Zhu, Zheng Qu, Peter Richtarik, and Yang Yuan. Even faster accelerated coordinate descent using non-uniform sampling. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1110–1119, New York, New York, USA, 20–22 Jun 2016. PMLR.
2. Michel Baes. Estimate sequence methods: extensions and approximations. Technical Report Optimization-Online 2372, Université Catholique de Louvain, August 2009.
3. Sébastien Bubeck, Yin Tat Lee, and Mohit Singh. A geometric alternative to Nesterov's accelerated gradient descent. Technical report, Microsoft Research, 2015. arXiv:1506.08187[math.OC].
4. Chih-Chung Chang and Chih-Jen Lin. LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, pages 2:27:1–27:27, 2011. `http://www.csie.ntu.edu.tw/cjlin/libsvmtools/datasets`.
5. Shixiang Chen, Shiqian Ma, and Wei Liu. Geometric descent method for convex composite minimization. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 636–644. Curran Associates, Inc., 2017.
6. Andrew Cotter, Ohad Shamir, Nati Srebro, and Karthik Sridharan. Better mini-batch algorithms via accelerated gradient methods. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 1647–1655. Curran Associates, Inc., 2011.
7. Jelena Diakonikolas and Lorenzo Orecchia. The approximate duality gap technique: A unified theory of first-order methods. *SIAM Journal on Optimization*, 29(1):660–689, 2019.
8. Dmitriy Drusvyatskiy, Maryam Fazel, and Scott Roy. An optimal first order method based on optimal quadratic averaging. *SIAM Journal on Optimization*, 28(1):251–271, 2018.
9. Olivier Fercoq and Peter Richtárik. Accelerated, parallel, and proximal coordinate descent. *SIAM Journal on Optimization*, 25(4):1997–2023, 2015. 10.1137/130949993.

10. Kimon Fountoulakis and Rachael Tappenden. A flexible coordinate descent method. *Computational Optimization and Applications*, 70(2):351–394, 2018.
11. Saeed Ghadimi and Guanghui Lan. Optimal stochastic approximation algorithms for strongly convex stochastic composite optimization I: A generic algorithmic framework. *SIAM Journal on Optimization*, 22(4):1469–1492, 2012. doi:10.1137/110848876.
12. Martin Jaggi, Virginia Smith, Martin Takac, Jonathan Terhorst, Sanjay Krishnan, Thomas Hofmann, and Michael I Jordan. Communication-e□cient distributed dual coordinate ascent. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3068–3076. Curran Associates, Inc., 2014.
13. Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 315–323. Curran Associates, Inc., 2013.
14. Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 7–9 May 2015, San Diego, USA, 2014. arXiv:1412.6980.
15. Hongzhou Lin, Julien Mairal, and Zaid Harchaoui. A universal catalyst for first-order optimization. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3384–3392. Curran Associates, Inc., 2015.
16. Zhaosong Lu and Lin Xiao. On the complexity analysis of randomized block-coordinate descent methods. *Mathematical Programming*, 152:615–642, 2015. doi:10.1007/s10107-014-0800-2.
17. Chenxin Ma, Martin Jaggi, Frank E. Curtis, Nathan Srebro, and Martin Takáč. An accelerated communication-e□cient primal-dual optimization framework for structured machine learning. Technical report, Lehigh University, USA, 2017. arXiv:1711.05305 [math.OC].
18. Chenxin Ma, Virginia Smith, Martin Jaggi, Michael Jordan, Peter Richtarik, and Martin Takac. Adding vs. averaging in distributed primal-dual optimization. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1973–1982, Lille, France, 07–09 Jul 2015. PMLR.
19. Yu Nesterov. E□ciency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.
20. Yurii Nesterov. A method for solving the convex programming problem with convergence rate $o(1/k^2)$. *Dokl. Akad. Nauk SSSR*, 269(3):543–547, 1983.
21. Yurii Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*, volume 87 of *Applied Optimization*. Springer (Originally published by Kluwer Academic Publishers), 2004. doi:10.1007/978-1-4419-8853-9.
22. Yurii Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103(1):127–152, 2005. doi:10.1007/s10107-004-0552-5.
23. Yurii Nesterov. Gradient methods for minimizing composite objective function. CORE Discussion Paper 2007/76, Université Catholique de Louvain, 2007.
24. Yurii Nesterov. Accelerating the cubic regularization of newton's method on convex problems. *Mathematical Programming*, 112(1):159–181, 2008.
25. Yurii Nesterov. Gradient methods for minimizing composite functions. *Mathematical Programming*, 140(1):125–161, 2013. doi:10.1007/s10107-012-0629-5.
26. Atsushi Nitanda. Stochastic proximal gradient descent with acceleration techniques. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 1574–1582. Curran Associates, Inc., 2014.
27. Brendan O'donoghue and Emmanuel Candes. Adaptive restart for accelerated gradient schemes. *Foundations of computational mathematics*, 15(3):715–732, 2015.
28. Peter Richtárik and Martin Takàč. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming*, 144(1-2):1–38, 2014.
29. Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

30. Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162(1-2):83–112, 2017. doi:10.1007/s10107-016-1030-6.
31. Shai Shalev-Shwartz and Tong Zhang. Accelerated mini-batch stochastic dual coordinate ascent. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 378–385. Curran Associates, Inc., 2013.
32. R. Tappenden, P. Richtárik, and J. Gondzio. Inexact coordinate descent: Complexity and preconditioning. *J Optim Theory Appl*, 170:144–176, 2016.
33. Rachael Tappenden, Peter Richtárik, and Jacek Gondzio. Inexact coordinate descent: Complexity and preconditioning. *Journal of Optimization Theory and Applications*, 170(1):144–176, 2016.

A Comparison of a UES and Nesterov's ES

Here we briefly compare the definition of an underestimate sequence with Nesterov's definition of an estimate sequence. The original definition of an ES only applied to smooth functions that satisfy Assumption 1 so we restrict our discussion to this case. Moreover, we will assume that the same sequence $\{\alpha_k\}_{k=0}^{\infty}$ is used when discussing a UES and an ES. Consider the following definition.

Definition 2 (Definition 2.2.1 in [21]) A pair of sequences $\{\varphi_k^N(x)\}_{k=0}^{\infty}$ and $\{\lambda_k^N\}_{k=0}^{\infty}$ $\lambda_k^N \geq 0$ is called an estimate sequence of function $f(x)$ if

(i) $\lambda_k^N \to 0$; and
(ii) for any $x \in \mathbb{R}^n$ and all $k \geq 0$ we have $\varphi_k^N(x) \leq (1-\lambda_k^N) f(x) + \lambda_k^N \varphi_0^N(x)$.

The definition is general and does not say anything about convergence. With this in mind, Nesterov's ES is coupled with the following lemma.

Lemma 10 (Lemma 2.2.1 in [21]) *If for some sequence $\{x_k\}$ we have*

$$f(x_k) \leq (\varphi_k^N)^* := \min_{x \in \mathbb{R}^n} \varphi_k^N(x), \tag{57}$$

then $f(x_k) - f^* \leq \lambda_k^N (\varphi_0^N(x^*) - f^*) \to 0$.

The first observation to make is the clear difference between f and $\varphi_k(x)$ for a UES and an ES. In particular, for a UES, at *every iteration* it holds that $\varphi_k(x) \leq F(x)$ (4) so every $\varphi_k(x)$ ($k \geq 0$) is a global underestimate (global lower bound) of the objective function *for all* $x \in \mathbb{R}^n$. However, for an ES, the function $\varphi_k^N(x)$ is not necessarily a global upper bound for f and it is necessarily not a global lower bound for f. What must hold is that, at iteration k, the minimizer of the approximation function $\varphi_k^N(x)$ must be at least as large as $f(x_k)$ (the value of the objective function at the current point).

A second major differences that one observes is as follows. If an algorithm generates a series of sequences that form a UES (satisfying Definition 1), then (assuming $\sum_{k=0}^{\infty} \alpha_k = \infty$), the algorithm is guaranteed to converge (see Proposition 1). On the other hand, if an algorithm generates a series of sequences that form an ES (satisfying Definition 2), there is no such algorithm convergence guarantee. This statement is made concrete by considering the next lemma and the text that follows it.

Lemma 11 (Lemma 2.2.2 in [21]) *Assume that (1) f satisfies Assumption 1, (2) $\varphi_0^N(x)$ is an arbitrary function on $\mathbb{R}^n$, (3) $\{y_k\}_{k=0}^{\infty}$ is an arbitrary sequence in $\mathbb{R}^n$, (4) $\{\alpha_k\}_{k=0}^{\infty}$ with $\alpha_k \in (0,1)$ and $\sum_{k=0}^{\infty} \alpha_k = \infty$ and (5) $\lambda_0^N = 1$. Then the pair of sequences $\{\varphi_k^N(x)\}_{k=0}^{\infty}$ and $\{\lambda_k^N\}_{k=0}^{\infty}$ recursively defined by*

$$\lambda_{k+1}^N = (1-\alpha_k)\lambda_k^N, \tag{58}$$

$$\varphi_{k+1}^N(x) = (1-\alpha_k)\varphi_k^N(x) + \alpha_k \left[f(y_k) + \langle \nabla f(y_k), x - y_k \rangle + \frac{\mu}{2} \|x - y_k\|^2 \right] \tag{59}$$

is an estimate sequence.

Combining (36), with (33) and (16) shows that $\varphi_{k+1}(x)$ in (36) is equivalent to $\varphi^N_{k+1}(x)$ in (59) and therefore, *the construction in this work is an estimate sequence* ($\varphi_{k+1}(x) \equiv \varphi^N_{k+1}(x)$). However, we will now show that the construction does not satisfy (57), and therefore, even though the iterates generated by SUESA/ASUESA form an estimate sequence, Lemma 10 *cannot* be used to prove convergence of SUESA/ASUESA.

Lemma 12 (Lemma 2.2.3 in [21]) *Let* $\varphi^N_0(x) = (\varphi^N_0) + \frac{\gamma^N_0}{2} \|x - v^N_0\|^2$. *Then the process described in Lemma 11 preserves the canonical form of functions* $\{\varphi^N_k(x)\}_{k=0}^{\infty}$:

$$\varphi^N_{k+1}(x) = (\varphi^N_{k+1}) + \frac{\gamma^N_k}{2} \|x - v^N_{k+1}\|^2, \tag{60}$$

where the sequences $\{\gamma^N_k\}_{k=0}^{\infty}$, $\{v^N_k\}_{k=0}^{\infty}$ *and* $\{(\varphi^N_k)\}_{k=0}^{\infty}$ *are defined as follows:*

$$\gamma^N_{k+1} = (1-\alpha_k)\gamma^N_k + \alpha_k\mu, \tag{61}$$

$$v^N_{k+1} = \frac{1}{\gamma^N_{k+1}}\left[(1-\alpha_k)\gamma^N_k v^N_k + \alpha_k\mu y_k - \alpha_k \nabla f(y_k)\right], \tag{62}$$

$$\begin{aligned}(\varphi^N_{k+1}) &= (1-\alpha_k)(\varphi^N_k) + \alpha_k f(y_k) - \frac{\alpha_k^2}{2\gamma^N_{k+1}}\|\nabla f(y_k)\|^2 \\ &\quad + \frac{\alpha_k(1-\alpha_k)\gamma^N_k}{\gamma^N_{k+1}}\left[\frac{\mu}{2}\|y_k - v^N_k\|^2 + \langle \nabla f(y_k), v^N_k - y_k\rangle\right]\end{aligned} \tag{63}$$

Note that in (60) the numerator in front of the norm term is γ^N_k, while in (37) it is μ. Substituting $\gamma^N_k = \mu$ into (61) gives $\gamma^N_{k+1} = \gamma^N_k = \mu$ so setting $\gamma^N_0 = \mu$ ensures γ^N_k is fixed for all $k \geq 0$ in Lemma 12. Now, using $\gamma^N_{k+1} = \gamma^N_k = \mu$ in (62), and recalling the form of a long step shows that $v^N_{k+1} \equiv v_{k+1}$ in (25). It remains to observe that (38) (combined with (16)) is equivalent to (63).

Thus, the difference between the construction in this work and the construction in [21] comes down to the minimizer and minimal value of $\varphi^N_0(x)$. For an ES, it must hold that $f(x_0) \leq (\varphi^N_0)$, and the initialization of scheme (2.2.6) in [21], as well as the proof of Theorem 2.2.1, explicitly mentioned the use of the choice $v_0 = x_0$ for Nesterov's method. However, note that other choices of v_0 can still provide the equality $(\varphi^N_0) = f(x_0)$, which means that the minimal value of the first element in the ES would be unchanged, i.e., $f(x_0)$, and the minimizer would be shifted from x_0 to other points. On the other hand, this contrasts with SUESA/ASUESA, where it is required that $\varphi_0(x) \leq f(x)$ and so they are initialized with $v_0 = x_0^{++}$ and $\varphi_0 = f(x_0) - \frac{1}{2\mu}\|\nabla f(x_0)\|^2 \leq f(x_0)$; see (35).

Finally, note that Definition 1 also holds for composite functions. While Definition 2 only holds for smooth functions, Nesterov has extended the ES framework to the composite setting; see Section 4 in [24] and Section 4 in [25]. Moreover, the relationship between the OQA method and an ES is discussed Appendix A in [8].

B Comparison of a UES and the study [7]

In [7], the authors proposed a general scheme for the analysis of first-order methods. For the strongly convex cases (both smooth and nonsmooth), there are some similarities between the methods in [7] and our proposed methods, but we stress that the approaches in this work are inherently different from those in [7]. To be more specific, Table 3 summarizes the iterates of ASUESA, and compares them with the iterates (for the smooth and strongly convex setting) in [7]. The similarities and differences between the corresponding two studies are described as follows:

1. The β_ks are different for each method, and if the problem is ill-conditioned (large κ), then the β_ks are close to each other ($\approx 1 - \frac{1}{\sqrt{\kappa}}$).

2. The y_ks have the same update structure, but because the β_ks are di□erent, this results in *di erent updates* y_k (later we will see that the v_ks and x_ks are also di□erent as the algorithms progress).
3. The x_{k+1}s are identical in *structure*. However, again the iterates x_{k+1} are *di erent* because the β_ks, and subsequently y_ks, are *di erent* for both methods.
4. The v_{k+1}s are di□erent both in structure and clearly in their values.
5. The x_ks in our proposed study form part of an underestimate sequence, which guarantees the natural stopping criteria, i.e., $f(x_k) - \varphi_k$ goes to zero in a linear rate.

Table 3 Comparison of the iterates

Iterates	ASUESA (current study)	Diakonikolas and Orecchia[2019]
β_k	$\frac{1}{1 + \sqrt{1/\kappa}}$	$1 - \frac{\sqrt{4\kappa + 1} - 1}{2\kappa}$
y_k	$\beta_k x_k + (1 - \beta_k)v_k$	$\beta_k x_k + (1 - \beta_k)v_k$
x_{k+1}	$y_k - \frac{1}{L} \nabla f(y_k)$	$y_k - \frac{1}{L} \nabla f(y_k)$
v_{k+1}	$(1 - \alpha_k)v_k + (\alpha_k)(y_k - \frac{1}{\mu} \nabla f(y_k))$	$\sum a_i(-\mu y_i - \mu A^{(k+1)} \nabla f(y_i))$