A GENERALIZED ACCELERATED COMPOSITE GRADIENT METHOD: UNITING NESTEROV’S FAST GRADIENT METHOD AND FISTA

MIHAI I. FLOREA* AND SERGIY A. VOROBOYV*

Abstract. We demonstrate that the augmented estimate sequence framework unites the most popular primal first-order schemes for large-scale problems: the Fast Gradient Method (FGM) and the Fast Iterative Shrinkage Thresholding Algorithm (FISTA). We further showcase the flexibility of the augmented estimate sequence by deriving a Generalized Accelerated Composite Gradient Method endowed with monotonicity alongside a versatile line-search procedure. The new method surpasses both FGM and FISTA in terms of robustness and usability. In particular, it is guaranteed to converge without requiring any quantitative prior information on the problem. Additional information, if available, leads to an improvement in performance at least on par with the state-of-the-art. We support our findings with simulation results.

Key words. estimate sequence, Fast Gradient Method, FISTA, monotone, line-search, composite objective, large-scale optimization

AMS subject classifications. 90C06, 68Q25, 90C25

1. Introduction. Numerous large-scale convex optimization problems have recently emerged in a variety of fields, including signal and image processing, statistical inference, computer vision, and machine learning. Often, little is known about the actual structure of the objective function. Therefore, optimization algorithms used in solving such problems can only rely (e.g. by means of callback functions) on specific black-box methods, called oracle functions [12]. The term “large-scale” refers to the tractability of certain computational primitives (see also [16]). In the black-box setting, it means that the oracle functions of large-scale problems usually only include scalar functions and operations that resemble first-order derivatives.

The introduction of Nesterov’s Fast Gradient Method (FGM) [13] has rendered large-scale applications, especially those with non-strongly convex objectives, practical to solve with sufficient accuracy. FGM requires that the objective be differentiable with Lipschitz gradient. Many optimization problems, particularly inverse problems in fields such as sparse signal processing, linear algebra, matrix and tensor completion, and digital imaging (see [3, 4, 6, 18, 21] and references therein), have a composite structure. In these composite problems, the objective \( F \) is the sum of a function \( f \) with Lipschitz gradient (Lipschitz constant \( L_f \)) and a simple but possibly non-differentiable regularizer \( \Psi \). The regularizer \( \Psi \) embeds constraints by being infinite outside the feasible set. Often, \( L_f \) is not known in advance. Composite problem oracle functions are scalar \( f(x) \) and \( \Psi(x) \), as well as gradient \( \nabla f(x) \) and proximal operator \( \text{prox}_{\tau \Psi}(x) \).

To address composite problems, Nesterov has devised an Accelerated Multistep Gradient Scheme (AMGS) [15]. This method updates a Lipschitz constant estimate (LCE) at every iteration using a subprocess commonly referred in the literature as “line-search” [2, 18]. The generation of a new iterate (advancement phase of an iteration) and line-search are interdependent and cannot be executed in parallel. AMGS also utilizes only gradient-type oracle functions \( \nabla f(x) \) and \( \text{prox}_{\tau \Psi}(x) \). In many applications, including compressed sensing (e.g. LASSO [23]) and many classification

*Department of Signal Processing and Acoustics, Aalto University, Espoo, Finland (mihai.florea@aalto.fi, sergiy.vorobyov@aalto.fi).
tasks (e.g., $l_1$-regularized logistic regression), the evaluation of $\nabla f(x)$ is more computationally expensive than $f(x)$. An alternative to AMGS that uses $f(x)$ calls in line-search has been proposed by Beck and Teboulle [2] in the form of the Fast Iterative Shrinkage-Thresholding Algorithm (FISTA). FISTA also benefits from having line-search decoupled from advancement. However, FISTA is unable to decrease the LCE at run-time. A strongly convex extension of FISTA, which we designate as FISTA Chambolle-Pock (FISTA-CP), has been recently proposed in [6], but without line-search.

FGM was derived using the estimate sequence [14]. This flexible framework was adapted in [15] to also include AMGS. FISTA-CP (as well as FISTA when the objective is non-strongly convex) is identical in form to FGM, and can be viewed as an extension of FGM for composite objectives. However, the convergence analyses of FISTA-CP and FISTA do not appear to involve the estimate sequence. Consequently, new features of FGM cannot be directly incorporated into FISTA and FISTA-CP. For instance, both FGM and FISTA-CP lack a line-search procedure. Recently, Nesterov has proposed in [17] a line-search variant of FGM, albeit only for non-strongly convex objectives. Neither the derivation nor convergence analysis have been provided, but can be readily obtained using the estimate sequence framework. Similar efforts have been made to add fully adaptive (proper) line-search to FISTA. For the non-strongly convex case, a sophisticated extension was proposed in [20], with a technical derivation based on the mathematical constructs of [2]. However, through partial adoption of the estimate sequence, i.e., relating FISTA to “constant step scheme I” in [14] and AMGS, we have arrived at a similar but simpler proper line-search for FISTA [8], but again only in the non-strongly convex scenario.

In [7], we have introduced the augmented estimate sequence framework and used it to derive the Accelerated Conjugate Gradient Method (ACGM), which incorporates by design a fully adaptive line-search procedure. ACGM has the convergence guarantees of FGM, the best among primal first-order methods, while being as broadly applicable as AMGS. In addition, FISTA-CP and FISTA, along with proper line-search extensions in [8] and [20], are particular cases of ACGM [7]. However, to accommodate infeasible start, we have imposed restrictions on the input parameters. Variants of FGM (e.g. “constant step scheme III” in [14]) exist that are guaranteed to converge only when the starting point is feasible, and thus do not correspond to any instance of ACGM.

In this paper, we generalize the augmented estimate sequence framework and derive a generalization of ACGM that encompasses both FGM and FISTA/FISTA-CP, along with their most common variants. We further showcase the flexibility and power of the augmented estimate sequence framework by endowing ACGM with monotonicity alongside its adaptive line-search procedure. Monotonicity is a desirable property, particularly when dealing with proximal operators that lack a closed form expression or other kinds of inexact oracles [1, 6]. By preventing overshoots, monotonicity also leads to a more stable and predictable convergence rate. The resulting generalized ACGM is able to converge even when no quantitative knowledge of the problem is available. It is thus superior to FGM and FISTA in terms of flexibility and usability. We support our theoretical findings with simulation examples.

1.1. Assumptions and notation. Consider composite optimization problems of the form

$$\min_{x \in \mathbb{R}^n} F(x) \triangleq f(x) + \Psi(x),$$
where $\mathbf{x} \in \mathbb{R}^n$ is a vector of $n$ optimization variables, and $F$ is the objective function. The constituents of the objective $F$ are the convex differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ and convex lower semicontinuous regularizer function $\Psi : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$. Function $f$ has Lipschitz gradient (Lipschitz constant $L_f > 0$) and strong convexity parameter $\mu_f \geq 0$ while $\Psi$ has strong convexity parameter $\mu_\Psi \geq 0$, entailing that objective $F$ has strong convexity parameter $\mu = \mu_f + \mu_\Psi$. The feasible set $\mathcal{D} \subseteq \mathbb{R}^n$ is convex and closed. Constraints are enforced by making $\Psi$ infinite outside $\mathcal{D}$.

Apart from the above properties, nothing is assumed known about functions $f$ and $\Psi$, which can only be accessed in a black-box fashion [12] by querying oracle functions $f(\mathbf{x})$, $\nabla f(\mathbf{x})$, $\Psi(\mathbf{x})$, and $\text{prox}_{\tau \Psi}(\mathbf{x})$, with arguments $\mathbf{x} \in \mathbb{R}^n$ and $\tau > 0$.

The proximal operator $\text{prox}_{\tau \Psi}(\mathbf{x})$ is given by

$$
\text{prox}_{\tau \Psi}(\mathbf{x}) \overset{\text{def}}{=} \arg\min_{\mathbf{z} \in \mathbb{R}^n} \left( \Psi(\mathbf{z}) + \frac{1}{2\tau} \|\mathbf{z} - \mathbf{x}\|^2_2 \right), \quad \mathbf{x} \in \mathbb{R}^n, \quad \tau > 0.
$$

Central to our derivation are generalized parabolae, quadratic functions whose Hessians are multiples of the identity matrix. We refer to the strongly convex ones simply as parabolae, of the form

$$
\psi : \mathbb{R}^n \to \mathbb{R}, \quad \psi(\mathbf{x}) \overset{\text{def}}{=} p + \frac{\gamma}{2} \|\mathbf{x} - \mathbf{v}\|^2_2, \quad \mathbf{x} \in \mathbb{R}^n,
$$

where $\gamma > 0$ denotes the curvature, $\mathbf{v}$ is the vertex, and $p$ is a constant.

For conciseness, we introduce the generalized parabola expression $Q_{f,\gamma,y}(\mathbf{x})$ as

$$
Q_{f,\gamma,y}(\mathbf{x}) \overset{\text{def}}{=} f(y) + \langle \nabla f(y), \mathbf{x} - y \rangle + \frac{\gamma}{2} \|\mathbf{x} - y\|^2_2, \quad \mathbf{x}, \mathbf{y} \in \mathbb{R}^n, \quad \gamma \geq 0.
$$

The proximal gradient operator $T_L(\mathbf{y})$ [15] can be expressed succinctly using (4) as

$$
T_L(\mathbf{y}) \overset{\text{def}}{=} \arg\min_{\mathbf{x} \in \mathbb{R}^n} (Q_{f,\gamma,y}(\mathbf{x}) + \Psi(\mathbf{x})), \quad \mathbf{y} \in \mathbb{R}^n,
$$

where $L > 0$ is a parameter corresponding to the inverse of the step size. Within the scope of this work, the left-hand side of (5) does not need functional parameters. Operator $T_L(\mathbf{y})$ can be evaluated in terms of oracle functions as

$$
T_L(\mathbf{y}) = \text{prox}_{\frac{1}{L} \Psi} \left( \mathbf{y} - \frac{1}{L} \nabla f(\mathbf{y}) \right), \quad \mathbf{y} \in \mathbb{R}^n.
$$

The composite gradient [15] is given by

$$
g_L(\mathbf{y}) \overset{\text{def}}{=} L \left( \mathbf{y} - T_L(\mathbf{y}) \right), \quad \mathbf{y} \in \mathbb{R}^n, \quad L > 0.
$$

We also define the relaxed supporting generalized parabola $\mathcal{R}_{L,y}(\mathbf{x})$ of objective $F$ at point $\mathbf{y}$ using inverse step size $L$ as

$$
\mathcal{R}_{L,y}(\mathbf{x}) \overset{\text{def}}{=} F(T_L(\mathbf{y})) + \frac{1}{2L} \|g_L(\mathbf{y})\|^2_2 + \langle g_L(\mathbf{y}), \mathbf{x} - \mathbf{y} \rangle + \frac{\mu}{2} \|\mathbf{x} - \mathbf{y}\|^2_2, \quad \mathbf{x} \in \mathbb{R}^n.
$$
2. Generalizing ACGM.

2.1. Nesterov’s first order method design pattern. Nesterov’s FGM and AMGS adhere to the design pattern outlined in Algorithm 1 (early variant discussed in [7]). This pattern will form the scaffolding of our generalized ACGM.

Algorithm 1 takes as input the starting point \( x_0 \), function \( \psi_0 \) and, if the Lipschitz constant is not known in advance, an initial LCE \( L_0 > 0 \). As we shall see later on, \( \psi_0 \) is the initial estimate function within the generalized augmented sequence framework (Subsection 2.3). The pattern in Algorithm 1 fits within the family of majorization-minimization algorithms. In line 5 of Algorithm 1, the main iterate is given by the minimum of \( u_{k+1}(x) \), a local upper bound on the objective. This upper bound is uniquely determined by an auxiliary point \( y_{k+1} \). Alongside the main iterate, the algorithm maintains an estimate function \( \psi_{k+1} \), obtained from the previous one by adding a global lower bound \( w_{k+1}(x) \) weighted by \( a_{k+1} \) (line 6 of Algorithm 1). The current LCE \( L_{k+1} \), weight \( a_{k+1} \), and auxiliary point \( y_{k+1} \) are computed using algorithm specific methods \( S, F_a, \) and \( F_y \), respectively (lines 2, 3, and 4 of Algorithm 1). These methods take as parameters the state of the algorithm, given by current values of the main iterate, LCE, weight, and estimate function.

Algorithm 1 A design pattern for Nesterov’s first-order accelerated algorithms

1: for \( k = 0, \ldots, K - 1 \) do
2: \( L_{k+1} = S(x_k, \psi_k, L_k) \)
3: \( a_{k+1} = F_a(\psi_k, L_{k+1}) \)
4: \( y_{k+1} = F_y(x_k, \psi_k, a_{k+1}) \)
5: \( x_{k+1} = \arg\min_{x \in \mathbb{R}^n} u_{k+1}(x) \)
6: \( \psi_{k+1}(x) = \psi_k(x) + a_{k+1}w_{k+1}(x) \)
7: end for

2.2. FGM Estimate Sequence. When the objective function is strongly convex, many first-order schemes, including the non-accelerated fixed-point methods, guarantee linear convergence of iterates to the optimal point. When the problem is non-strongly convex, the optimization landscape may contain a high-dimensional subspace of very low curvature in the vicinity of the set of optimal points. In this case, the convergence of iterates remains a difficult open problem [5]. For instance, Nesterov has provided in [14] an ill-conditioned quadratic problem where the convergence of iterates to an optimal point is intractable for all first-order schemes of a certain structure.

Hence, we choose to measure convergence using the image space distance (ISD), which is the distance between the objective values at iterates and the optimal value. The decrease rate of an upper bound on the ISD gives the convergence guarantee (provable convergence rate). The estimate sequence framework follows naturally from the formulation of such guarantees. Specifically, we interpret the image space distance upper bound (ISDUB), provided by Nesterov for FGM in [14], as

\[
A_k(F(x_k) - F(x^*)) \leq A_0(F(x_0) - F(x^*)) + \frac{\gamma_0}{2}\|x_0 - x^*\|^2_2, \quad x^* \in X^*, \quad k \geq 0.
\]

Here, the convergence guarantee is given by the sequence \( \{A_k\}_{k \geq 0} \) with \( A_0 \geq 0 \) and \( A_k > 0 \) for all \( k \geq 1 \). The right-hand side of (9) is a weighted sum between the initial ISD and the corresponding domain space term (DST), with weights given by
$A_0$ and $\gamma_0$. In the derivation of FGM, the weights are constrained as $A_0 > 0$ and $\gamma_0 \geq A_0 \mu$. When the starting point $x_0$ is not guaranteed to be feasible, $A_0$ must be zero. For AMGS, $\gamma_0$ is fixed as 1 while for original ACGM [7], to prevent $A_0$ from being unbounded above, we have enforced $\gamma_0 \leq 1$. Given that our current aim is to provide a generic framework, we impose no restrictions on the weights, apart from $A_0 \geq 0$ and $\gamma_0 > 0$. The former restriction follows from the convexity of $F$ while the latter is required by the estimate sequence, along with its augmented variant, as we shall demonstrate in the sequel.

The ISDUB expression can be rearranged to take the form

$$A_k F(x_k) \leq H_k,$$

where

$$H_k \overset{\text{def}}{=} (A_k - A_0) F(x^*) + A_0 F(x_0) + \frac{\gamma_0}{2} \|x_0 - x^*\|_2^2$$

is the highest upper bound that can be placed on weighted objective values $A_k F(x_k)$ to satisfy (9).

The value of $H_k$ depends on the optimal value $F(x^*)$, which is an unknown quantity. The estimate sequence provides a computable, albeit more stringent, replacement for $H_k$. It is obtained as follows. The convexity of the objective implies the existence of a sequence $\{W_k\}_{k \geq 1}$ of convex global lower bounds on $F$, namely

$$F(x) \geq W_k(x), \quad x \in \mathbb{R}^n, \quad k \geq 1.$$

By substituting the optimal value terms $F(x^*)$ in (10) with $W_k(x^*)$, we obtain $\mathcal{H}_k$, a lower bound on $H_k$, given by

$$\mathcal{H}_k \overset{\text{def}}{=} (A_k - A_0) W_k(x^*) + A_0 F(x_0) + \frac{\gamma_0}{2} \|x^* - x_0\|_2^2, \quad k \geq 0.$$

This still depends on an unknown quantity. However, $\mathcal{H}_k$ can be viewed as the value of an estimate function, taken at an optimal point $x^*$. The estimate functions $\psi_k(x)$, $k \geq 0$ are defined as functional extensions of $\mathcal{H}_k$, namely

$$\psi_k(x) \overset{\text{def}}{=} (A_k - A_0) W_k(x) + A_0 F(x_0) + \frac{\gamma_0}{2} \|x - x_0\|_2^2, \quad x \in \mathbb{R}^n, \quad k \geq 0.$$

Note that the first estimate function $\psi_0$ does not contain a lower bound term. Therefore, it is not necessary to define $W_0$. The collection of estimate functions $\{\psi_k(x)\}_{k \geq 0}$, is referred to as the estimate sequence.

The estimate function optimum value, given by

$$\psi_k^* \overset{\text{def}}{=} \min_{x \in \mathbb{R}^n} \psi_k(x), \quad k \geq 0,$$

is guaranteed to be lower than $\mathcal{H}_k$, since

$$\psi_k^* = \min_{x \in \mathbb{R}^n} \psi_k(x) \leq \psi_k(x^*) = \mathcal{H}_k, \quad k \geq 0.$$

As such, $\psi_k^*$ provides the sought after computable replacement of $H_k$. Note that if the lower bounds $W_k$ are linear, the estimate functions are generalized parabolae with the curvature given by $\gamma_0$. In this case, the existence of $\psi_k^*$ (in non-degenerate cases) is conditioned by $\gamma_0 > 0$, explaining the assumption made in (9).
Thus, it suffices to maintain the augmented estimate sequence property, given by
\[
A_k F(x_k) \leq \psi^*_k, \quad k \geq 0,
\]
to satisfy the ISDUB expression (9). The proof follows from the above definitions as follows:
\[
A_k F(x_k) \overset{(17)}{\leq} \psi_k^* \overset{(16)}{\leq} \psi_k(x^*) = \mathcal{H}_k \overset{(12)}{\leq} H_k, \quad k \geq 0.
\]

The interval between the maintained upper bound \(\psi^*_k\) and the highest allowable bound \(H_k\) contains \(\mathcal{H}_k\). As the iterates \(x_k\) approach an optimum point \(x^*\), the distance between \(\psi^*_k\) and \(\mathcal{H}_k\) vanishes. However, if lower bounds \(W_k\) are not tight at the optimum, the distance between \(\mathcal{H}_k\) and \(H_k\) may not diminish at all. In these situations, the estimate sequence property (17) may not be possible to maintain, due to its stringency. Thus, by introducing the augmented estimate sequence, we address these shortcomings and create a unifying framework applicable to composite problems.

2.3. Generalizing the Augmented Estimate Sequence. Nesterov’s FGM and AMGS were designed by induction to preserve the estimate sequence property in any algorithmic state. Such a worst-case analysis alleviates the need for an algorithm to maintain the estimate sequence optimum. Therefore, it is possible to construct an algorithm by introducing unknown quantities in the estimate sequence optimum. In this work, to allow the distance between the maintained upper bounds and the theoretical limit to vanish when the algorithm approaches the region of optimal points, regardless of how tight the lower bounds are, we forcibly close the gap between \(\mathcal{H}_k\) and \(H_k\). Namely, we define the augmented estimate functions as
\[
\psi'_k(x) \overset{\text{def}}{=} \psi_k(x) + H_k - \mathcal{H}_k, \quad k \geq 0,
\]
with \(\{\psi'_k(x)\}_{k \geq 0}\) being the augmented estimate sequence. We expand definition (19) as
\[
\psi'_k(x) = \psi_k(x) + (A_k - A_0)(F(x^*) - W_k(x^*)).
\]
The augmented estimate sequence property is thus given by
\[
A_k F(x_k) \leq \psi'^*_k.
\]

2.4. Towards an algorithm. In the design pattern outlined in Algorithm 1, we set the lower bounds to be supporting generalized parabolae, namely
\[
w_{k+1}(x) = R_{L_{k+1} + \mu_k, y_{k+1}}(x), \quad x \in \mathbb{R}^n, \quad k \geq 0.
\]
To ensure that the supporting generalized parabolae are valid lower bounds on the objective \(F\), we enforce that all auxiliary points and LCEs obey the descent condition, stated as
\[
f(z_{k+1}) \leq Q_{f, L_{k+1}, y_{k+1}}(z_{k+1}), \quad k \geq 0,
\]
where
\[
z_{k+1} = T_{L_{k+1}}(y_{k+1}).
\]
The proof of this sufficient condition can be found in [7, Lemma 2], with $x_{k+1}$ replaced by $z_{k+1}$.

By combining the estimate function update in line 6 of Algorithm 1 with the estimate function definition (14), we obtain a recursion rule for the lower bounds $W_{k+1}(x)$ in the form of

$$W_{k+1}(x) = \frac{(A_k - A_0)W_k(x) + a_{k+1}w_{k+1}(x)}{A_{k+1} - A_0}, \quad k \geq 0.$$  \hfill (25)

For functions $W_{k+1}(x)$ to be valid lower bounds on the objective, regardless of the sign or tightness of lower bounds $w_{k+1}(x)$, the following must hold for all $k \geq 0$:

$$a_{k+1} \geq 0,$$  \hfill (26)

$$A_{k+1} = A_k + a_{k+1},$$  \hfill (27)

$$W_{k+1}(x) = \frac{1}{A_{k+1} - A_0} \sum_{i=1}^{k+1} (a_i w_i(x)).$$  \hfill (28)

From definition (14), we have that the initial estimate function is a parabola, given by

$$\psi_0(x) = A_0 F(x_0) + \frac{\gamma_0}{2} \| x - x_0 \|^2.$$  \hfill (29)

From (22) and (26), line 6 of Algorithm 1 further ensures that estimate functions at every iteration are parabolic. We write the estimate functions and the augmented estimate functions, for all $k \geq 0$, as

$$\psi_k(x) = \psi_k^* + \frac{\gamma_k}{2} \| x - v_k \|^2,$$  \hfill (30)

$$\psi_k'(x) = \psi_k'^* + \frac{\gamma_k}{2} \| x - v_k \|^2,$$  \hfill (31)

with

$$\psi_k'^* = \psi_k^* + (A_k - A_0)(F(x^*) - W_k(x^*)).$$  \hfill (32)

Following Nesterov’s design procedure in [14], we select by induction functions $S$, $F_a$, and $F_y$ so as to preserve the augmented estimate sequence property (17). First, for $k = 0$, the augmented estimate sequence property trivially holds, since $\psi_0'^* = A_0 F(x_0)$. Next, we assume that (17) holds for an arbitrary $k \geq 0$ and we devise expressions for $S$, $F_a$, and $F_y$ that ensure that (17) holds for $k + 1$, irrespective of the state parameter values. A sufficient condition for the preservation of (17) as the algorithm progresses is that the augmented estimate sequence gap, defined as

$$\Gamma_k \overset{\text{def}}{=} A_k F(x_k) - \psi_k'^*, \quad k \geq 0,$$  \hfill (33)

is non-increasing. This gap can be written as

$$\Gamma_k \overset{(32)}{=} A_k (F(x_k) - F(x^*)) + (A_k - A_0) W_k(x^*) + A_0 F(x^*) - \psi_k^*$$

$$\overset{(30)}{=} A_k (F(x_k) - F(x^*)) + \frac{\gamma_k}{2} \| x^* - v_k \|^2 -$$

$$- \psi_k(x^*) + A_0 F(x^*) + (A_k - A_0) W_k(x^*)$$

$$\overset{(14)}{=} A_k (F(x_k) - F(x^*)) + \frac{\gamma_k}{2} \| v_k - x^* \|^2 -$$

$$- A_0 (F(x_0) - F(x^*)) - \frac{\gamma_0}{2} \| x_0 - x^* \|^2, \quad k \geq 0.$$  \hfill (34)
We introduce the gap sequence \( \{ \Delta_k \}_{k \geq 0} \) in the form
\[
\Delta_k \overset{\text{def}}{=} A_k (F(x_k) - F(x^*)) + \frac{\gamma_k}{2} \| v_k - x^* \|^2_2, \quad k \geq 0.
\]

The augmented estimate sequence gaps can be expressed more succinctly as
\[
\Gamma_k = \Delta_k - \Delta_0, \quad k \geq 0.
\]

Hence, the variation of the two sequences is identical, with the only difference being that the augmented estimate sequence gap is constrained to be zero initially. The sufficient condition becomes
\[
\Delta_{k+1} \leq \Delta_k, \quad k \geq 0.
\]

The enforced descent condition (23) can be equivalently expressed in terms of composite objective values as
\[
F(z_{k+1}) \leq F(z_k) \leq Q_{f,L_k+1,y_k+1}(z_{k+1}) + \Psi(z_{k+1}), \quad x \in \mathbb{R}^n.
\]

In [7, Theorem 3] we have shown that if we choose \( x_{k+1} = z_{k+1} \), we can build a method that maintains a monotone gap sequence. In this work we allow for point \( x_{k+1} \) to be better than \( z_{k+1} \) in the form of the following result.

**Theorem 1.** If at iteration \( k \geq 0 \) we have
\[
F(x_{k+1}) \leq F(z_{k+1}) \leq Q_{f,L_k+1,y_k+1}(z_{k+1}) + \Psi(z_{k+1}),
\]
then
\[
\Delta_{k+1} + A_k + B_k \leq \Delta_k,
\]
where subexpressions \( A_k, B_k \), and the reduced composite gradient \( G_k \) are defined as
\[
A_k \overset{\text{def}}{=} \frac{1}{2} \left( \frac{A_{k+1}}{L_{k+1} + \mu \Psi} - \frac{a_{k+1}^2}{\gamma_{k+1}} \right) \| g_{L_{k+1} + \mu \Psi} (y_{k+1}) \|^2_2,
\]
\[
B_k \overset{\text{def}}{=} \frac{1}{\gamma_{k+1}} (G_k, A_k \gamma_{k+1} x_k + a_{k+1} \gamma_{k} v_k - (A_k \gamma_{k+1} + a_{k+1} \gamma_{k}) y_{k+1}),
\]
\[
G_k \overset{\text{def}}{=} g_{L_{k+1} + \mu \Psi} (y_{k+1}) - \mu y_{k+1}.
\]

**Proof.** The estimate sequence update in line 6 of Algorithm 1 along with (30), (6), and (48) gives, e.g. through successive differentiation, update rules for the estimate sequence curvatures and vertices, for all \( k \geq 0 \), as follows:
\[
\gamma_{k+1} = \gamma_k + a_{k+1} \mu,
\]
\[
v_{k+1} = \frac{1}{\gamma_{k+1}} (\gamma_k v_k + a_{k+1} (L_{k+1} + \mu \Psi) z_{k+1} - a_{k+1} (L_{k+1} - \mu f) y_{k+1}).
\]

The descent condition assumption implies lower bound property (22). Let the tightness of this lower bound be given by the residual \( R(x) \) as
\[
R(x) \overset{\text{def}}{=} F(x) - \mathcal{R}_{L_{k+1} + \mu \Psi,y_{k+1}}(x), \quad x \in \mathbb{R}^n, \quad k \geq 0.
\]
From (44) and (26) we have that $\gamma_{k+1} \geq \gamma_k$. Along with $A_k R(x_k) + a_{k+1} R(x^*) \geq 0$, we obtain, using the proof mechanics of [7, Theorem 3], that

$$\gamma_{k+1} \geq \frac{\gamma_k}{2} \|v_{k+1} - x^*\|^2 - \frac{\gamma_k}{2} \|v_k - x^*\|^2 + A_k + B_k.$$  

Combining $F(x_{k+1}) \leq F(z_{k+1})$ with (47) gives the desired result.

Theorem 1 provides a simple sufficient condition for the monotonicity of the gap sequence, regardless of the algorithmic state, given by the following relations, for all $k \geq 0$:

$$y_{k+1} = F_y(x_k, \psi_k, A_k, A_{k+1}) = \frac{A_k \gamma_{k+1} x_k + a_{k+1} \gamma_k v_k}{A_k \gamma_{k+1} + a_{k+1} \gamma_k},$$

(48)

$$L_{k+1} + \mu \Psi a_{k+1}^2 \leq A_{k+1} \gamma_{k+1}.$$  

(49)

The latter, combined with the non-negativity of the weights (26), yields

$$a_{k+1} \leq E(\gamma_k, A_k, L_{k+1}), \quad k \geq 0,$$

(50)

where expression $E(\gamma_k, A_k, L_{k+1})$ is given by

$$E(\gamma_k, A_k, L_{k+1}) \triangleq \gamma_k + A_k \mu + \frac{\sqrt{(\gamma_k + A_k \mu)^2 + 4(L_{k+1} - \mu f) A_k \gamma_k}}{2(L_{k+1} - \mu f)}.$$  

(51)

The cumulative weight $A_k$ in (27) gives the convergence guarantee in (9). To provide the best guarantees, we enforce equality in (50), namely

$$a_{k+1} = F_a(\psi_k, A_k, L_{k+1}) = E(\gamma_k, A_k, L_{k+1}).$$  

(52)

Interestingly, all of the above results do not rely on a specific form of the local upper bound $u_{k+1}(x)$, as long as assumption (39) holds for all $k \geq 0$. We want our algorithm to converge as fast as possible while maintaining the monotonicity property, expressed as

$$F(x_{k+1}) \leq F(x_k), \quad k \geq 0.$$  

(53)

Then, without further knowledge of the objective function, (24), (39), and (53) suggest a simple expression of the upper bound in the form of

$$u_{k+1}(x) = \min \{Q_f, L_{k+1}, y_{k+1}(x) + \Psi(x), F(x_k) + \sigma_{\{x_k\}}(x)\}, \quad k \geq 0,$$

(54)

where $\sigma_X$ is the indicator function [19] of set $X$, given by

$$\sigma_X(x) = \begin{cases} 0, & x \in X, \\ +\infty, & \text{otherwise}. \end{cases}$$  

(55)

For determining the LCE, we select the backtracking line-search method $S_A$ employed by AMGS [15] and the original ACGM [7]. The search parameters comprise the LCE increase rate $r_u > 1$ and the LCE decrease rate $0 < r_d < 1$. Search terminates when the line-search stopping criterion (LSSC) in (23) is satisfied.
2.5. Putting it all together. We have thus determined a search strategy $\mathcal{S}_A$, initial estimate function $v_0$ in (29), upper bounds $u_{k+1}(x)$ in (54), lower bounds $w_{k+1}(x)$ in (22), function $F_b$ in (52), and function $F_y$ in (48). Substituting these expressions in the design pattern outlined in Algorithm 1, we can write down a generalization of ACGM in estimate sequence form, as listed in Algorithm 2.

Non-monotone generalized ACGM can be obtained by enforcing $x_{k+1} = z_{k+1}$ for all $k \geq 0$, accomplished by replacing line 17 of Algorithm 2 with

$$x_{k+1} := \hat{z}_{k+1}.$$  

Algorithm 2 Generalized monotone ACGM in estimate sequence form

1: function ACGM($x_0$, $L_0$, $\mu_f$, $\mu_\Psi$, $A_0$, $\gamma_0$, $r_u$, $r_d$, $K$) \> Initialization
2: $v_0 = x_0$, $\mu = \mu_f + \mu_\Psi$
3: for $k = 0, \ldots, K - 1$ do \> Main loop
4: \hspace{1em} $\hat{L}_{k+1} := r_dL_k$
5: \hspace{2em} loop
6: \hspace{3em} $\hat{\alpha}_{k+1} := \frac{1}{2(L_{k+1} - \mu_f)} \left( \gamma_k + A_k\mu + \sqrt{(\gamma_k + A_k\mu)^2 + 4(L_{k+1} - \mu_f)A_k\gamma_k} \right)$
7: \hspace{3em} $\hat{A}_{k+1} := A_k + \hat{\alpha}_{k+1}$
8: \hspace{3em} $\hat{\gamma}_{k+1} := \gamma_k + \hat{\alpha}_{k+1}\mu$
9: \hspace{3em} $y_{k+1} := L_k \gamma_{k+1} + \frac{1}{A_k\gamma_{k+1}}(A_k\gamma_{k+1}x_k + \hat{\alpha}_{k+1}\gamma_k v_k)$
10: \hspace{3em} $\hat{z}_{k+1} := \text{prox}_{\frac{1}{r_u_L_{k+1}}} \left( y_{k+1} - \frac{1}{L_{k+1}} \nabla f(\hat{y}_{k+1}) \right)$
11: \hspace{2em} if $f(\hat{z}_{k+1}) \leq Q_jL_{k+1}, \hat{y}_{k+1}(\hat{z}_{k+1})$ then
12: \hspace{3em} Break from loop
13: \hspace{2em} else
14: \hspace{3em} $\hat{L}_{k+1} := r_uL_{k+1}$
15: \hspace{2em} end if
16: \hspace{2em} end loop
17: \hspace{2em} $x_{k+1} := \text{arg\,min}\{F(\hat{z}_{k+1}), F(x_k)\}$
18: \hspace{2em} $v_{k+1} := \frac{1}{\gamma_k} (\gamma_k v_k + \hat{\alpha}_{k+1}(\hat{L}_{k+1} + \mu_\Psi)\hat{z}_{k+1} - \hat{\alpha}_{k+1}(\hat{L}_{k+1} - \mu_f)\hat{y}_{k+1})$
19: \hspace{2em} $L_{k+1} := \hat{L}_{k+1}$, $A_{k+1} := \hat{A}_{k+1}$, $\gamma_{k+1} := \hat{\gamma}_{k+1}$
20: \hspace{2em} end for
21: return $x_K$ \> Output
22: end function

3. Complexity analysis.

3.1. Worst-case convergence guarantees. Algorithm 2 maintains the convergence guarantee in (9) explicitly at run-time as state variable $A_k$. Moreover, if sufficient knowledge of the problem is available, it is possible to formulate a worst-case convergence guarantee before running the algorithm.

For our analysis, we will need to define a number of curvature-related quantities, namely the local inverse condition number $q_{k+1}$ for all $k \geq 0$, the worst-case LCE $L_u$, and the worst-case inverse condition number $q_u$, given by

$$q_{k+1} \overset{\text{def}}{=} \frac{\mu}{L_{k+1} + \mu_\Psi}, \quad L_u \overset{\text{def}}{=} \max\{r_uL_f, r_dL_0\}, \quad q_u \overset{\text{def}}{=} \frac{\mu}{L_u + \mu_\Psi}.$$  

The worst-case convergence guarantees for generalized ACGM are stated in following theorem.
Theorem 2. If $\gamma_0 \geq A_0 \mu$, the generalized ACGM algorithm generates a sequence $\{x_k\}_{k \geq 1}$ that satisfies

$$F(x_k) - F(x^*) \leq \min \left\{ \frac{4}{(k+1)^2}, (1 - \sqrt{q_u})^{k-1}\right\} (L_u - \mu_f) \bar{\Delta}_0, \quad k \geq 1.$$  \hspace{1cm} (58)

where

$$\bar{\Delta}_0 \overset{\text{def}}{=} \frac{\Delta_0}{\gamma_0} = \frac{A_0}{\gamma_0} (F(x_0) - F(x^*)) + \frac{1}{2} \|x_0 - x^*\|_2^2.$$  \hspace{1cm} (59)

Proof. The estimate function curvature is given by

$$\gamma_k = \gamma_0 + \sum_{i=1}^{k} a_i \mu = \gamma_0 - A_0 \mu + A_k \mu, \quad k \geq 0.$$  \hspace{1cm} (60)

The non-negativity of the weights (26) implies that $\gamma_k \geq \gamma_0$ for all $k \geq 0$. Combined with (52), we have

$$A_{k+1} \geq A_k + \frac{\gamma_0}{2(L_{k+1} - \mu_f)} + \sqrt{\frac{\gamma_0^2}{4(L_{k+1} - \mu_f)^2} + \frac{A_k \gamma_0}{(L_{k+1} - \mu_f)}}, \quad k \geq 0.$$  \hspace{1cm} (61)

As we can see from Algorithm 2, scaling $A_0$ and $\gamma_0$ by a fixed factor does not alter the behavior of generalized ACGM. Additionally, $\gamma_0$ is guaranteed to be non-zero. To simplify calculations, we introduce the normalized convergence guarantees $\bar{A}_k \overset{\text{def}}{=} A_k / \gamma_0$ for all $k \geq 0$.

Regardless of the outcome of individual line-search calls, the growth of the normalized accumulated weights obeys

$$\bar{A}_{k+1} \geq \bar{A}_k + \frac{1}{2(L_u - \mu_f)} + \sqrt{\frac{1}{4(L_u - \mu_f)^2} + \frac{\bar{A}_k}{(L_u - \mu_f)}}, \quad k \geq 0.$$  \hspace{1cm} (62)

Taking into account that $A_0 \geq 0$, we obtain by induction that

$$\bar{A}_k \geq \frac{(k+1)^2}{4(L_u - \mu_f)}, \quad k \geq 1.$$  \hspace{1cm} (63)

From assumption $\gamma_0 \geq A_0 \mu$, (60) implies $\gamma_k \geq A_k \mu$ for all $k \geq 0$. Hence

$$a_{k+1}^2 A_{k+1} \overset{(52)}{=} \frac{\gamma_{k+1}}{(L_{k+1} + \mu_f) A_{k+1}} \geq \frac{\mu}{L_{k+1} + \mu_f} = q_{k+1} \geq q_u, \quad k \geq 0.$$  \hspace{1cm} (64)

Since $A_0 \geq 0$, we have that $\bar{A}_1 \geq \frac{1}{L_u - \mu_f}$. By induction, it follows that

$$\bar{A}_k \geq \frac{1}{L_u - \mu_f} (1 - \sqrt{q_u})^{-(k-1)}, \quad k \geq 1.$$  \hspace{1cm} (65)

Substituting (63) and (65) in (9) completes the proof. $\blacksquare$
3.2. Wall-clock time units. So far, we have measured the theoretical performance of algorithms in terms of convergence guarantees (including the worst-case ones) indexed in iterations. This does not account for the complexity of individual iterations. In [7], we have introduced a new measure of complexity, the wall-clock time unit (WTU), to compare optimization algorithms more reliably. We thus distinguish between two types of convergence guarantees. One is the previously used iteration convergence guarantee, indexed in iterations and a new computational convergence guarantee, indexed in WTU.

The WTU is a measure of running time in a shared memory parallel scenario. The computing environment consists of a small number of parallel processing units (PPU). Each PPU may be a parallel machine itself. The number of parallel units is considered sufficient to compute any number of independent oracle functions simultaneously. The shared-memory system does not impose constraints on parallelization, namely, it is uniform memory access (UMA) [10] and it is large enough to store the arguments and results of oracle calls for as long as they are needed.

In order to compare algorithms based on a unified benchmark, in [7] we have assumed that \( f \) and \( \nabla f \) require 1 WTU each while all other operations are negligible and amount to 0 WTU. In this work, we generalize the analysis. We attribute finite non-negative costs \( t_f \), \( t_g \), \( t_\Psi \), and \( t_p \) to \( f(x) \), \( \nabla f(x) \), \( \Psi(x) \), and \( \text{prox}_\tau \Psi(x) \), respectively. However, since we are dealing with large-scale problems, we maintain the assumption that element-wise vector operations, including scalar-vector multiplications, vector additions, and inner products, have negligible complexity when compared to oracle functions and assign a cost of 0 WTU to each. Synchronization of PPUs also incurs no cost. Consequently, when computed in isolation, an objective function value \( F(x) \) call costs \( t_F = \max\{t_f, t_\Psi\} \), ascribable to separability, while a proximal gradient operation costs \( t_T = t_g + t_p \), due to computational dependencies.

3.3. Per-iteration complexity. We measure this complexity in WTU on the shared memory system described in the previous subsection and consider a parallel implementation involving speculative execution [10].

The advancement phase of a generalized ACGM iteration consists of one proximal gradient step (line 10 of Algorithm 2). Hence, every iteration has a base cost of \( t_T = t_g + t_p \). LSSC and the monotonicity condition (MC) in line 17 of Algorithm 2 can be evaluated in parallel with subsequent iterations. Both rely on the computation of \( f(\hat{z}_{k+1}) \), which in the worst case requires \( \lceil t_f/t_T \rceil \) dedicated PPUs. In addition, MC may need up to \( \lceil t_\Psi/t_T \rceil \) PPUs.

Backtracks stall the algorithm in a way that cannot be alleviated by parallelization or intensity reduction. Therefore, it is desirable to make them a rare event. Assuming that the local curvature of \( f \) varies around a fixed value, this would mean that \( \log(r_u) \) should be significantly larger than \( -\log(r_d) \). With such a parameter choice, the algorithm can proceed from one iteration to another by speculating that backtracks do not occur at all. Let the current iteration be indexed by \( k \). If the LSSC of iteration \( k \) fails, then the algorithm discards all the state information pertaining to all iterations made after \( k \), reverts to iteration \( k \), and performs the necessary computation to correct the error. We consider that a mis-prediction incurs a detection cost \( t_d \) and a correction cost \( t_c \). LSSC requires the evaluation of \( f(\hat{z}_{k+1}) \) and incurs a detection cost of \( t_d = t_f \). A backtrack entails recomputing \( \hat{y}_{k+1} \), yielding an LSSC \( t_c = t_f \) correction time.

Overshoots are assumed to occur even less often. Similarly, the algorithm proceeds by speculating that MC always passes and defaults to (56). Hence, MC has \( t_d = t_F \), due to its dependency on \( \Psi(\hat{z}_{k+1}) \), but once the algorithmic state of iteration \( k \) has
been restored, no additional oracle calls are needed, leading to $t_c = 0$. MC and LSSC can be fused into a single condition, giving rise to the scenarios outlined in Table 1. Note that if LSSC fails, MC is not evaluated.

| LSSC passed | MC passed | Max\{tf, tf\} |
|-------------|-----------|--------------|
| LSSC failed | tf + tg + tp | N/A          |

Table 1
Algorithm stall time in WTU based on the outcome of LSSC and MC

For non-monotone generalized ACGM, each backtrack adds $tf + t_T$ WTU to a base iteration cost of $t_T$. A comparison to other methods employing line-search is shown in Table 2.

| LSSC passed | MC passed | MC failed |
|-------------|-----------|-----------|
| FISTA       | $tg + tp$ | $tg + tp$ |
| AMGS        | $tg + 2tp$| $tg + tp$ |
| ACGM        | $tg + 2tp$| $tf + tg + tp$ |

Table 2
Per-iteration cost of FISTA, AMGS, and generalized ACGM in the non-monotone setting

4. Extrapolated form.

4.1. Monotonicity and extrapolation. In the original ACGM [7], the auxiliary point can be obtained from two successive main iterates through extrapolation. Interestingly, this property is preserved for any value of the inverse step size. We show in the following how monotonicity alters this property and bring generalized monotone ACGM to a form in which the auxiliary point is an extrapolation of state variables. First, we observe that estimate sequence vertices can be obtained from main iterates through extrapolation, namely

\begin{equation}
\mathbf{v}_{k+1} = \mathbf{x}_k + \frac{A_{k+1}}{a_{k+1}} (z_{k+1} - x_k), \quad k \geq 0.
\end{equation}

The proof does not require any conditions on $A_0$ or $\gamma_0$ and is thus the same as the one in [7, Lemma 5]. Combined with the auxiliary point update (48) it leads to

\begin{equation}
\mathbf{y}_{k+1} = \frac{1}{A_k \gamma_k + a_{k+1} \gamma_k} \left( \frac{A_k}{a_k} \mathbf{x}_k + \frac{A_k}{a_k} \mathbf{z}_k + \left( a_{k+1} \gamma_k - \frac{A_k}{a_k} \right) x_{k-1} \right), \quad k \geq 1.
\end{equation}

Depending on the outcome of the update in line 17 of Algorithm 2, we distinguish two situations.

If MC passes at iteration $k - 1$ ($F(z_k) \leq F(x_{k-1})$), then

\begin{equation}
\mathbf{y}_{k+1} = (1 + b_k) \mathbf{z}_k - b_k \mathbf{x}_{k-1} = \mathbf{x}_k + b_k (\mathbf{z}_k - \mathbf{x}_{k-1}),
\end{equation}

where, for brevity, we define extrapolation factor $b_k$ and subexpression $\omega_k$ as

\begin{equation}
b_k \overset{\text{def}}{=} \left( \frac{A_k}{a_k} - 1 \right) \omega_k, \quad \omega_k \overset{\text{def}}{=} \frac{a_{k+1} \gamma_k}{A_k \gamma_k + a_{k+1} \gamma_k}.
\end{equation}
If the algorithm overshoots \((F(z_k) > F(x_{k-1}))\), then \(x_k = x_{k-1}\), which leads to

\[
y_{k+1} = b'_k z_k - (b'_k - 1)x_{k-1} = x_k + b'_k (z_k - x_{k-1}),
\]

where the extrapolation factor \(b'_k\) is given by

\[
b'_k \equiv \left( \frac{A_k}{a_k} \right) \omega_k.
\]

Expressions (68) and (70) lead to the following auxiliary point extrapolation rule:

\[
y_{k+1} = x_k + \beta_k (z_k - x_{k-1}), \quad k \geq 1,
\]

where

\[
\beta_k = \begin{cases} 
  b_k, & x_k = z_k \\
  b'_k, & x_k = x_{k-1}
\end{cases}, \quad k \geq 1.
\]

Until this point we have assumed that the first iteration \(k = 0\) does not use auxiliary point extrapolation rule (72). To write generalized ACGM in a form similar to monotone FISTA (MFISTA [1]) and the monotone version of FISTA-CP [6], we define the vertex extrapolation factor in (66) as

\[
t_k \equiv \frac{A_k}{a_k}, \quad k \geq 1.
\]

As long as \(k \geq 1\), (52) enables us to obtain a recursion rule for the vertex extrapolation factor that does not depend on weights \(a_k\) and \(A_k\), given by

\[
t_{k+1}^2 + t_{k+1} (q_k t_k^2 - 1) - \frac{L_{k+1} + \mu \Psi}{L_k + \mu \Psi} t_k^2 = 0, \quad k \geq 1, \quad \mu \geq 0.
\]

Subexpression \(\omega_k\) and auxiliary point extrapolation factor \(\beta_k\) can also be written as

\[
\omega_k = \frac{1 - q_{k+1} t_{k+1}}{(1 - q_{k+1}) t_{k+1}}, \quad k \geq 1,
\]

\[
\beta_k = \begin{cases} 
  (t_k - 1) \omega_k, & x_k = z_k \\
  t_k \omega_k, & x_k = x_{k-1}
\end{cases}, \quad k \geq 1.
\]

Note that subexpression \(\omega_k\) contains only recent information whereas \(\beta_k\) needs only to access the state of the preceding iteration.

For simplicity, we wish to extend update rules (72), (75), and (77) to the first iteration \(k = 0\). The missing parameters follow naturally from this extension. First, \(t_0\) can be obtained by setting \(k = 0\) in (75) as

\[
t_0 = \sqrt{\frac{t_1^2 - t_1}{L_0 + \mu \Psi - t_1 q_0}} \quad \text{(74)} = \sqrt{\frac{A_1 - a_1}{(L_0 + \mu \Psi) A_0 - a_1 q_0}} \quad \text{(49)} = \sqrt{\frac{(L_0 + \mu \Psi) A_0}{\gamma_0}}.
\]

Next, we introduce a “phantom iteration” \(k = -1\) with the main iterate as the only state parameter. We set \(x_{-1} \equiv x_0\) so that any value of \(\beta_0\) will satisfy (72). For brevity, we obtain \(\beta_0\) from expression (77) with \(k = 0\). We do not define \(a_0\). Instead, extrapolation factors \(b_k\) and \(b'_k\) from (69) and (71) can be computed when \(k = 0\) by replacing \(A_0/a_0\) with \(t_0\) expression (78).
Thus, with initialization (78) and recursion (75), we have completely defined the vertex extrapolation factor sequence \( \{t_k\}_{k \geq 0} \), and derived from it the auxiliary extrapolation factor expression (77). Now, we do not need to maintain weight sequences \( \{a_k\}_{k \geq 1} \) and \( \{A_k\}_{k \geq 0} \). We simplify generalized ACGM further by noting that, to produce the auxiliary point, extrapolation rule (72) depends on three vector parameters. However, it is not necessary to store both \( z_k \) and \( x_{k-1} \) across iterations. To address applications where memory is limited, we only maintain the difference term \( d_k \), given by
\[
d_k = (t_k - 1_{\{z_k\}}(x_k))(z_k - x_{k-1}), \quad k \geq 0,
\]
where \( 1_X \) denotes the membership function of set \( X \), namely
\[
1_X(x) = \begin{cases} 1, & x \in X \\ 0, & x \notin X \end{cases}.
\]
Extrapolation rule (72) becomes
\[
y_{k+1} = x_k + \omega_k d_k, \quad k \geq 0.
\]

The above modifications yield a form of generalized ACGM based on extrapolation, which we list in Algorithm 3. To obtain a non-monotone algorithm, it suffices to replace line 17 of Algorithm 3 with (56).

We stress that while Algorithms 2 and 3 carry out different computations, they are mathematically equivalent with respect to the main iterate sequence \( \{x_k\}_{k \geq 0} \). Oracle calls and their dependencies in Algorithm 3 are also identical to those in Algorithm 2. Therefore the per-iteration complexity is the same.

### 4.2. Retrieving the convergence guarantee.

In Algorithm 2, the convergence guarantee in expression (9) is obtained directly from a single state variable \( A_k \). For Algorithm 3, we distinguish two scenarios. The most common one is outlined in the following lemma.

**Lemma 3.** If \( \gamma_0 > A_0 \mu \), then
\[
A_k = \frac{(\gamma_0 - A_0 \mu) t_k^2}{(L_k + \mu \Psi)(1 - q_k t_k^2)}, \quad k \geq 1.
\]

**Proof.** Equality in (49) gives
\[
(L_k + \mu \Psi)A_k = \gamma_k t_k^2, \quad k \geq 1.
\]
Substituting curvature \( \gamma_k \) in (60) we obtain that
\[
(L_k + \mu \Psi)(1 - q_k t_k^2)A_k = (\gamma_0 - A_0 \mu) t_k^2.
\]
We prove by induction that
\[
t_k < \frac{1}{\sqrt{q_k}}, \quad k \geq 0.
\]
Condition \( \gamma_0 > A_0 \mu \) implies that \( t_0 < \frac{1}{\sqrt{q_0}} \). Next, we assume that (85) holds for an arbitrary \( k \geq 0 \). From (75) and the non-negativity of \( t_k+1 \) we have that
\[
t_{k+1} < \frac{L_{k+1} + \mu \Psi}{L_k + \mu \Psi} t_k^2 < \frac{1}{\sqrt{q_{k+1}}}.
\]
Thus, (85) holds for all \( k \geq 0 \). By dividing with the positive quantity \( 1 - q_k t_k^2 \) in (84), we get the desired result. \( \square \)
Algorithm 3 Generalized monotone ACGM in extrapolated form

1: function ACGM($x_0, L_0, \mu_f, \mu, A_0, \gamma_0, r_u, r_d, K$) 
2: \hspace{1cm} $x_{-1} = x_0, \ d_0 = 0$ \hspace{1cm} \triangleright Initialization 
3: $\mu = \mu_f + \mu, \ t_0 = \sqrt{\frac{(L_0 + \mu + \mu_f)A_0}{\gamma_0}}, \ q_0 = \frac{\mu}{L_0 + \mu}$ 
4: for $k = 0, \ldots, K-1$ do \hspace{1cm} \triangleright Main loop 
5: \hspace{1cm} $L_{k+1} := r_d L_k$ 
6: \hspace{1cm} loop 
7: \hspace{1.5cm} $\hat{q}_{k+1} := \frac{\mu}{L_{k+1} + \mu}$ 
8: \hspace{1.5cm} $\hat{t}_{k+1} := \frac{1}{2} \left(1 - q_k t_k^2 + \sqrt{(1 - q_k t_k^2)^2 + 4 \left(\frac{L_{k+1} + \mu + \mu_f}{L_0 + \mu} - q_k \right)^2}ight)$ 
9: \hspace{1.5cm} $\hat{y}_{k+1} := x_k + \frac{1 - \hat{q}_{k+1} \hat{t}_{k+1}}{(1 - \hat{q}_{k+1}) \hat{q}_{k+1}} d_k$ 
10: \hspace{1.5cm} $\hat{z}_{k+1} := \text{prox}_{\frac{1}{L_{k+1}}} \left(\hat{y}_{k+1} - \frac{1}{L_{k+1}} \nabla f(\hat{y}_{k+1})\right)$ 
11: if $f(\hat{z}_{k+1}) \leq Q_f L_{k+1} \hat{y}_{k+1} (\hat{z}_{k+1})$ then 
12: Break from loop 
13: else 
14: \hspace{1cm} $\hat{L}_{k+1} := r_u \hat{L}_{k+1}$ 
15: end if 
16: end loop 
17: $x_{k+1} := \arg\min \{F(\hat{z}_{k+1}), F(x_k)\}$ 
18: $d_{k+1} := (\hat{t}_{k+1} - 1) (\hat{z}_{k+1} - x_k)$ 
19: $L_{k+1} := \hat{L}_{k+1}, \ q_{k+1} := \hat{q}_{k+1}, \ t_{k+1} := \hat{t}_{k+1}$ 
20: end for 
21: return $x_K$ \hspace{1cm} \triangleright Output 
22: end function

Consequently, if $\gamma_0 > A_0 \mu$, the convergence guarantee can be derived directly from the state parameters without alterations to Algorithm 3.

4.3. Border-case. In the strongly convex case, $\gamma_0 = A_0 \mu$ is a valid parameter choice. In this situation, the state parameters of Algorithm 3 no longer contain information on the convergence guarantee as evidenced by the following result.

Lemma 4. If $\gamma_0 = A_0 \mu$, then $t_k = \frac{1}{\sqrt{q_k}}$ for all $k \geq 0$.

Proof. We have that $t_0 = \frac{1}{\sqrt{q_0}}$. Using (75), the rest of the proof follows by induction in the same manner as the proof of Lemma 3.

With this border-case parameter choice, Algorithm 3 can be brought to a simpler form. Lemma 4 implies that the auxiliary point extrapolation factor is given by

$$\beta_k = \frac{\sqrt{L_k + \mu} - 1 (z_k) (x_k) \sqrt{\mu}}{\sqrt{L_{k+1} + \mu} + \mu + \sqrt{\mu}}, \quad k \geq 0.$$ 

The sequence $\{t_k\}_{k \geq 0}$ does not store any relevant information and can be left out. This means that the convergence guarantee $A_k$ requires a dedicated update. Lemma 4 provides a simple recursion rule in the form of

$$A_{k+1} = \frac{1}{1 - \sqrt{q_{k+1}} A_k}, \quad k \geq 0.$$
Due to scaling invariance, we can select any pair \((A_0, \gamma_0)\) that is a positive multiple of \((1, \mu)\). For simplicity, we choose \(A_0 = 1\) and \(\gamma_0 = \mu\).

To reduce computational intensity, we modify subexpressions \(d_k\) and \(\omega_k\) as

\[
d_k = \left(\sqrt{L_k + \mu \Psi} - 1_{\{z_k\}}(\mathbf{x}_k)\sqrt{\mu}\right)(z_k - x_k - 1), \quad k \geq 0,
\]

\[
\omega_k = \frac{1}{\sqrt{L_{k+1} + \mu \Psi + \sqrt{\mu}}}, \quad k \geq 0.
\]

The local inverse condition number sequence \(\{q_k\}_{k \geq 0}\) does not appear in updates (87) and (88). Hence, it can also be abstracted away. The form taken by generalized ACGM in this border-case, after simplifications, is listed in Algorithm 4.

**Algorithm 4** Border-case ACGM in extrapolated form

1: function ACGM\((x_0, L_0, \mu_f, \mu_\Psi, r_u, r_d, K)\)
2: \(x_{-1} = x_0, \quad d_0 = 0, \quad A_0 = 1, \quad \mu = \mu_f + \mu_\Psi\) \hspace{1cm} \triangleright Initialization
3: for \(k = 0, \ldots, K-1\) do \hspace{1cm} \triangleright Main loop
4: \(\hat{L}_{k+1} := r_d L_k\)
5: \hspace{1cm} loop
6: \(\hat{y}_{k+1} := x_k + \frac{1}{\sqrt{L_{k+1} + \mu \Psi + \sqrt{\mu}}} d_k\)
7: \(\hat{z}_{k+1} := \text{prox}_{\frac{1}{\hat{L}_{k+1}} \Psi}\left(\hat{y}_{k+1} - \frac{1}{\hat{L}_{k+1}} \nabla f(\hat{y}_{k+1})\right)\)
8: \(\text{if } f(\hat{z}_{k+1}) \leq Q_{f, L_{k+1}}(\hat{z}_{k+1}) \text{ then} \)
9: \hspace{1cm} Break from loop
10: \(\text{else} \)
11: \hspace{1cm} \(\hat{L}_{k+1} := r_u \hat{L}_{k+1}\)
12: \hspace{1cm} end if
13: end loop
14: \(x_{k+1} := \arg\min_{x} \{F(\hat{z}_{k+1}), F(x)\}\)
15: \(d_{k+1} := \left(\sqrt{\hat{L}_{k+1} + \mu \Psi} - 1_{\{\hat{z}_{k+1}\}}(\mathbf{x}_{k+1})\sqrt{\mu}\right)(\hat{z}_{k+1} - x_k)\)
16: \(L_{k+1} := \hat{L}_{k+1}\)
17: \(A_{k+1} := \frac{\sqrt{L_{k+1} + \mu \Psi}}{\sqrt{L_{k+1} + \mu \Psi - \sqrt{\mu}}} A_k\) \hspace{1cm} \triangleright Optional
18: end for
19: return \(x_K\) \hspace{1cm} \triangleright Output
20: end function

A non-monotone variant can be obtained by replacing line 14 of Algorithm 4, with (56). The border-case iteration complexity matches the one of Algorithms 2 and 3.

5. Simulation results.

5.1. Benchmark setup. We have tested the variants of generalized ACGM introduced in this work against the methods considered at the time of writing to be the state-of-the-art on the problem class outlined in Subsection 1.1. The proposed methods included in the benchmark are non-monotone ACGM (denoted as plain ACGM), monotone ACGM (MACGM), and, for strongly-convex problems, border-case non-monotone ACGM (BACGM) as well as border-case monotone ACGM (BMACGM). The state-of-the-art methods are FISTA-CP, monotone FISTA-CP (MFISTA-CP) [6], AMGS [15], and FISTA with backtracking line-search (FISTA-BT) [2].
We have selected as test cases five synthetic instances of composite problems in the areas of statistics, inverse problems, and machine learning. Three are non-strongly convex: least absolute shrinkage and selection operator (LASSO) [23], non-negative least squares (NNLS), and $l_1$-regularized logistic regression (L1LR). The other two are strongly-convex: ridge regression (RR) and elastic net (EN) [24]. Table 3 lists the oracle functions of all above mentioned problems. Here, the sum softplus function

$$I(x), \text{ the element-wise logistic function } L(x), \text{ and the shrinkage operator } T_\tau(x) \text{ are, respectively, given by}$$

\begin{align}
I(x) &= \sum_{i=1}^{m} \log(1 + e^{x_i}), \quad L(x)_i = \frac{1}{1 + e^{-x_i}}, \quad i \in \{1, ..., m\}, \\
T_\tau(x)_j &= (|x_j| - \tau)_+ sgn(x_j), \quad j \in \{1, ..., n\}.
\end{align}

To attain the best convergence guarantees for AMGS, Nesterov suggests in [15] that all known global strong convexity be transferred to the simple function $\Psi$. When line-search is enabled, generalized ACGM also benefits slightly from this arrangement when $r_n > 1$ (Theorem 2). Without line-search, the convergence guarantees of generalized ACGM do not change as a result of strong convexity transfer, in either direction. Thus, for fair comparison, we have incorporated in $\Psi$ the strongly-convex quadratic regularization term for RR and EN problems. In the following, we describe in detail each of the five problem instances. All random variables are independent and identically distributed, unless stated otherwise.

**LASSO.** Real-valued matrix $A$ is of size $m = 500 \times n = 500$, with entries drawn from $\mathcal{N}(0, 1)$. Vector $b \in \mathbb{R}^m$ has entries sampled from $\mathcal{N}(0, 9)$. Regularization parameter $\lambda_1$ is 4. The starting point $x_0 \in \mathbb{R}^n$ has entries drawn from $\mathcal{N}(0, 1)$.

**NNLS.** Sparse $m = 1000 \times n = 10000$ matrix $A$ has approximately 10% of entries, at random locations, non-zero. The non-zero entries are drawn from $\mathcal{N}(0, 1)$ after which each column $j \in \{1, ..., n\}$ is scaled independently to have an $l_2$ norm of 1. Starting point $x_0$ has 10 entries at random locations all equal to 4 and the remainder zero. Vector $b$ is obtained from $b = Ax_0 + z$, where $z$ is standard Gaussian noise.

**L1LR.** Matrix $A$ has $m = 200 \times n = 1000$ entries sampled from $\mathcal{N}(0, 1)$, $x_0$ has exactly 10 non-zero entries at random locations, each entry value drawn from $\mathcal{N}(0, 225)$, and $\lambda_1 = 5$. Labels $y_i \in \{0, 1\}, i \in \{1, ..., m\}$ are selected with probability $\mathbb{P}(Y_i = 1) = L(Ax_i)$.

**RR.** Dimensions are $m = 500 \times n = 500$. The entries of matrix $A$, vector $b$, and starting point $x_0$ are drawn from $\mathcal{N}(0, 1)$, $\mathcal{N}(0, 25)$, and $\mathcal{N}(0, 1)$, respectively. Regularizer $\lambda_2$ is given by $10^{-3}(\sigma_{\text{max}}(A))^2$, where $\sigma_{\text{max}}(A)$ is the largest singular value of $A$.  

| Regularizer | $f(x)$ | $\Psi(x)$ | $\nabla f(x)$ | $\text{prox}_{\tau \Psi}(x)$ |
|-------------|--------|-----------|---------------|----------------------------|
| LASSO       | $\frac{1}{2}\|Ax - b\|^2$ | $\lambda_1 \|x\|_1$ | $A^T(Ax - b)$ | $T_{\tau \lambda_1}(x)$ |
| NNLS        | $\frac{1}{2}\|Ax - b\|^2$ | $\sigma_{\text{max}}(x)$ | $A^T(Ax - b)$ | $T_{\tau \lambda_1}(x)$ |
| L1LR        | $I(Ax) - y^T Ax$ | $\lambda_1 \|x\|_1$ | $A^T(L(Ax) - y)$ | $T_{\tau \lambda_1}(x)$ |
| RR          | $\frac{1}{2}\|Ax - b\|^2$ | $\frac{2}{\lambda_2} \|x\|_2^2$ | $A^T(Ax - b)$ | $T_{\tau \lambda_1}(x)$ |
| EN          | $\frac{1}{2}\|Ax - b\|^2$ | $\lambda_1 \|x\|_1 + \frac{1}{\lambda_2} \|x\|_2^2$ | $A^T(Ax - b)$ | $T_{\tau \lambda_1}(x)$ |
Matrix $A$ has $m = 1000 \times n = 500$ entries sampled from $N(0, 1)$. Starting point $x_0$ has 20 non-zero entries at random locations, each entry value drawn from $N(0, 1)$. Regularization parameter $\lambda_1$ is obtained according to [9] as $1.5 \sqrt{2 \log(n)}$ and $\lambda_2$ is the same as in RR, namely $10^{-3} (\sigma_{\text{max}}(A))^2$.

The Lipschitz constant $L_f$ is given by $(\sigma_{\text{max}}(A))^2$ for all problems except for L1LR where it is $\frac{1}{4} (\sigma_{\text{max}}(A))^2$. For strongly convex problems RR and EN, we have $\mu = \mu_f = \lambda_2$ and inverse condition number $q = \mu / (L_f + \mu_f) = 1/1001$.

To be able to benchmark against FISTA-CP and FISTA-BT, which lack proper line-search, we have set $L_0 = L_f$ for all tested algorithms, thus giving FISTA-CP and FISTA-BT an advantage over the proposed methods. To highlight the differences between ACGM and BACGM, we ran ACGM and MACGM with parameters $A_0 = 0$ and $\gamma_0 = 1$.

Despite the problems differing in structure, the oracle functions have the same computational costs. We consider one matrix-vector multiplication to cost 1 WTU. Consequently, for all problems, we have $t_f = 1$ WTU, $t_g = 2$ WTU. The regularizer has negligible cost, $t_\Psi = t_p = 0$ WTU.

The line-search parameters were selected according to the recommendation given in [3]. For AMGS and FISTA-BT we have $r_{\text{AMGS}} = r_{\text{FISTA}} = 2.0$ and $r_{\text{AMGS}} = 0.9$. The variants of generalized ACGM and AMGS are the only methods included in the benchmark that are equipped with fully adaptive line-search. We have decided to select $r_{\text{ACGM}}$ to ensure that ACGM and AMGS have the same overhead. We formally define the line-search overhead of method $M$, denoted by $\Omega^M$, as the average computational cost attributable to backtracks per WTU of advancement. Assuming that the LCEs hover around a fixed value (Subsection 3.3), we thus have that

\begin{align}
\Omega^\text{AMGS} &= \frac{(2t_g + t_p) \log(r_{\text{AMGS}})}{2(t_g + t_p) \log(r_{\text{AMGS}})}, \\
\Omega^\text{ACGM} &= \frac{(t_f + t_g + t_p) \log(r_{\text{ACGM}})}{(t_g + t_p) \log(r_{\text{ACGM}})}.
\end{align}

From (93) we have that $r_{\text{ACGM}} = (r_{\text{AMGS}})^{\frac{1}{2}}$, with no difference for border-case or monotone variants.

For measuring ISDs, we have computed beforehand an optimal point estimate $\hat{x}^*$ for each problem instance. Each $\hat{x}^*$ was obtained as the main iterate after running MACGM for 5000 iterations with parameters $A_0 = 0$, $\gamma_0 = 1$, $L_0 = L_f$ and aggressive search parameters $r_d = 0.9$ and $r_u = 2.0$.

5.2. Non-strongly convex problems. The convergence results for LASSO, NNLS, and LiLR are shown in Figure 1. The LCE variation during the first 200 WTU is shown in Figure 2. For NNLS, floating point precision was exhausted after 100 WTU and the LCE variation was only plotted to this point (Figure 2(b)). In addition, the average LCEs are listed in Table 4.

Both variants of ACGM outperform in iterations and especially in WTU the competing methods in each of these problem instances. Even though for LASSO and NNLS, the iteration convergence rate of AMGS is slightly better in the beginning (Figures 1(a) and 1(c)), AMGS lags behind afterwards and, when measured in terms of computational convergence rate, has the poorest performance among the methods tested (Figures 1(b), 1(d), and 1(f)). FISTA-BT produces the same iterates as FISTA-CP, as theoretically guaranteed in the non-strongly convex case for $L_0 = L_f$. 
Fig. 1. Convergence results of FISTA with backtracking (FISTA-BT), AMGS, FISTA-CP, monotone FISTA-CP (MFISTA-CP), non-monotone ACGM and monotone ACGM (MACGM) on the LASSO, NNLS, and L1LR non-strongly convex problems. Dots mark iterations preceding overshoots. At these iterations, the convergence behavior changes.
The overall superiority of ACGM and MACGM can be attributed to the effectiveness of line-search. Interestingly, ACGM manages to surpass FISTA-CP and MFISTA-CP even when the latter are supplied with the exact value of the global Lipschitz constant. This is because ACGM is able to accurately estimate the local curvature, which is often below $L_f$. For the L1LR problem, where the smooth part $f$ is not the square of a linear function, the local curvature is substantially lower than the global Lipschitz constant with LCEs hovering around one fifth of $L_f$ (Figure 2(c)). One would expect AMGS to be able to estimate local curvature as accurately as ACGM. This is does not happen due to AMGS’s reliance on a “damped relaxation condition” [15] line-search stopping criterion. For LASSO and NNSL the average LCE of AMGS is actually above $L_f$. ACGM has an average LCE that is more than 40% lower than AMGS on these problems whereas for L1LR the average is more than three times lower than AMGS. The difference between the LCE averages of ACGM and MACGM is negligible.

Indeed, monotonicity, as predicted, does not alter the overall iteration convergence rate and has a stabilizing effect. MACGM overshoots do have a negative but limited

---

**Table 4**

Average LCEs of line-search methods on LASSO, NNLS, and L1LR

| Problem | $L_f$  | Iterations | FISTA-BT | AMGS   | ACGM   | MACGM   |
|---------|-------|------------|----------|--------|--------|--------|
| LASSO   | 1981.98 | 2000       | 1981.98  | 2202.66| 1385.85| 1303.70|
| NNLS    | 17.17  | 50         | 17.17    | 19.86  | 14.35  | 13.54  |
| L1LR    | 518.79 | 200        | 518.79   | 246.56 | 80.76  | 79.12  |
impact on the computational convergence rate. We have noticed in our simulations that overshoots occur less often for larger problems, such as the tested instance of NNLS.

5.3. Strongly convex problems. The convergence results for RR and EN are shown in Figures 3(a), 3(b), 3(c), and 3(d). The LCE variation is shown in Figure 4 with LCE averages listed in Table 5.

Strongly convex problems have a unique optimum point and accelerated first-order schemes are guaranteed to find an accurate estimate of it in domain space (see [14] for detailed analysis). Along with Theorem 2, it follows that

\[
A_0(F(x_0) - F(\hat{x}^*)) + \frac{\gamma_0}{2}\|x_0 - \hat{x}^*\|^2 \simeq \Delta_0.
\]

Thus, we can display accurate estimates of ISDUBs in (9), of the form \(U_k \overset{\text{def}}{=} \frac{\Delta_0}{A_k}\), for methods that maintain convergence guarantees at runtime. These are shown in Figures 3(e) and 3(f) as upper bounds indexed in WTU.

For the smooth RR problem, the effectiveness of each algorithm tested is roughly given by the increase rate of the accumulated weights (Figures 3(a) and 3(c)). In iterations, AMGS converges the fastest. However, in terms of WTU usage, it is the least effective of the methods designed to deal with strongly convex objectives. The reasons are the high cost of its iterations, its low asymptotic rate compared to ACGM and FISTA-CP, and the stringency of its damped relaxation criterion that results in higher LCEs (on average) than ACGM (Figure 4(a) and Table 5). The computational convergence rate of BACGM is the best, followed by ACGM, FISTA-CP. This does not, however, correspond to the upper bounds (Figure 3(e)). While BACGM produces the largest accumulated weights \(A_k\), the high value of the ISD term in \(\Delta_0\) causes BACGM to have poorer upper bounds than ACGM, except for the first iterations. In fact, the effectiveness of BACGM on this problem is exceptional, partly due to the regularity of the composite gradients. This regularity also ensures monotonicity of BACGM, ACGM, and FISTA-CP. FISTA-BT does not exhibit linear convergence on this problem and it is even slower than AMGS after 500 WTU despite its lower line-search overhead and advantageous parameter choice \(L_0 = L_f\).

On the less regular EN problem, ACGM leads all other methods in terms of both iteration and computational convergence rates (Figures 3(b) and 3(d)). The advantage of ACGM, especially over BACGM, is accurately reflected in the upper bounds (Figure 3(f)). However, convergence is much faster than the upper bounds would imply. Even FISTA-BT has a competitive rate, due to the small number of iterations (150) needed for high accuracy results. The ineffectiveness of AMGS on this problem is mostly due to its high LCEs (Figure 4(b) and Table 5). The proposed ACGM and variants show comparable average LCEs. Here as well, monotonicity has a stabilizing effect and does not have a significant impact on the computational convergence rate.

6. Discussion. Our simulation results suggest that enforcing monotonicity in ACGM is generally beneficial in large-scale applications. It leads to a more predictable convergence rate and, provided that the number of overshoots per iteration is small, monotonicity has a negligible impact on the computational convergence rate as well. Our experimental results indicate that the frequency of overshoots generally decreases with problem size.
Fig. 3. Convergence results of FISTA with backtracking (FISTA-BT), AMGS, FISTA-CP, monotone FISTA-CP (MFISTA-CP), non-monotone ACGM, monotone ACGM (MACGM), border-case non-monotone ACGM (BACGM), and border-case monotone ACGM (BMACGM) on the RR and EN strongly-convex problems. Dots mark iterations preceding overshoots.
From a theoretical standpoint, the proposed method can be viewed as a unification of FGM and FISTA, in their most common forms. Specifically, the fixed-step variant ($L_k = L_f$ for all $k \geq 0$) of ACGM in extrapolated form (Algorithm 3) is equivalent to both the monotone and non-monotone variants of FISTA-CP with the theoretically optimal step size $\tau_{\text{FISTA-CP}} = \frac{1}{L_f}$. Moreover, when $\mu = 0$, non-monotone original fixed-step ACGM coincides with the original formulation of FISTA in [2]. Adding monotonicity yields MFISTA [1]. Also for $\mu = 0$, but without the fixed-step restriction, the original non-monotone ACGM in estimate sequence form reduces to the robust FISTA-like algorithm in [8], whereas in extrapolated form it is a valuable simplification of the method introduced in [20].

When dealing with differentiable objectives, we can assume without loss of generality that $\Psi(x) = 0$ for all $x \in \mathbb{R}^n$. In what follows, we consider generalized non-monotone fixed-step ACGM in estimate sequence form, unless stated otherwise. By substituting the local upper bound functions $u_{k+1}(x)$ at every iteration $k \geq 0$ with any functions that produce iterates satisfying the descent condition, which means in this context that

$$f(x_{k+1}) \leq f(y_{k+1}) - \frac{1}{2L_{k+1}}\|\nabla f(y_{k+1})\|_2^2,$$

where $x_{k+1}$ is given by line 5 of Algorithm 1, we obtain the “general scheme of optimal method” in [14]. Both the monotone and non-monotone variants adhere to this scheme. The correspondence between Nesterov’s notation in [14] and ours is, for all $k \geq 0$, given by:

$$\lambda_k^{\text{PGM}} = \frac{A^0}{A^k_{\text{ACGM}}}, \quad \phi_k^{\text{PGM}}(x) = \frac{1}{A^k_{\text{ACGM}}} \psi^{\text{ACGM}}_k(x), \quad x \in \mathbb{R}^n,$$

$$y_k^{\text{PGM}} = y_{k+1}^{\text{ACGM}}, \quad \alpha_k^{\text{PGM}} = \frac{\sigma_k^{\text{ACGM}}}{A^k_{\text{ACGM}}}, \quad \gamma_k^{\text{PGM}} = \frac{\gamma_k^{\text{ACGM}}}{A^k_{\text{ACGM}}}.$$

The remaining state parameters are identical.
Note that FGM makes the assumption that $A_0^{ACGM} > 0$, which is incompatible with the original specification of ACGM in [7]. With the above assumption, the non-monotone variant (Algorithm 2) is in fact identical to “constant step scheme Γ” in [14]. Similarly, the extrapolated form of fixed-step non-monotone ACGM (Algorithm 3) corresponds exactly to “constant step scheme II” in [14] while fixed-step border-case non-monotone ACGM (Algorithm 4) is identical to “constant step scheme III” in [14]. We note that in our simulation results, the ISD term in $\Delta_0$ is large compared to the DST and consequently the upper bounds are poorer in the border-case. This extends to the fixed-step setup and disputes the overemphasis in the literature (e.g. [11, 22]) on the border-case form. In fact, this form may constitute the poorest choice of parameters $A_0$ and $\gamma_0$ in many applications. Indeed, the worst-case results in Theorem 2 favor $A_0 = 0$.

The FGM variant in [17] is a particular case of non-monotone ACGM with variable step size (Algorithm 2) when the objective is non-strongly convex ($\mu = 0$) and the step size search parameters are set to $r_u^{ACGM} = 2$ and $r_d^{ACGM} = 0.5$. The notation correspondence is as follows:

$$
\begin{align*}
  x_{k+1,i}^{FGM} &= x_{k+1,i}^{ACGM}, \\
  y_{k,i}^{FGM} &= y_{k,i}^{ACGM}, \\
  a_{k,i}^{FGM} &= a_{k+1,i}^{ACGM}, \\
  2^i L_f &= L_k^{ACGM},
\end{align*}
$$

with all other parameters identical.

Thus, by relaxing the assumption that $A_0 = 0$, we have devised a generalized variant of ACGM that effectively encompasses FGM [14], with its recently introduced variant [17], as well as the original FISTA [2], including its adaptive step-size variants [8, 20], the monotone version MFISTA [1], and the strongly convex extension FISTA-CP [6]. A summary of how the above first-order methods relate to generalized ACGM is given in Table 6.

### Table 6

| Algorithm         | Smooth objective | Fixed step size | $\mu = 0$ | Restriction           | Non-monotone | Monotone | $A_0 = 0$ | $A_0 > 0$ |
|-------------------|------------------|-----------------|-----------|-----------------------|--------------|----------|-----------|-----------|
| FGM [14]          | yes              | yes             | no        | yes                   | no           | no       | yes       |           |
| FGM [17]          | yes              | no              | yes       | yes                   | no           | uncertain| uncertain |           |
| FISTA [2]         | no               | partial         | yes       | yes                   | no           | yes      | no        |           |
| MFISTA [1]        | no               | yes             | yes       | no                    | yes          | yes      | no        |           |
| FISTA-CP [6]      | no               | yes             | no        | no                    | no           | no       | no        |           |

Due to its adaptivity, generalized ACGM is not limited to the composite problem framework in Subsection 1.1. It is also guaranteed to converge on problems where the gradient of $f$ is not globally Lipschitz continuous. Constituent function $f$ needs to have Lipschitz gradient only in the area explored by the algorithm.

In terms of usability, generalized ACGM does not require a priori knowledge of Lipschitz constant $L_f$, or a lower estimate of it, beforehand. Thus, the proposed method can be utilized without any quantitative knowledge of the problem. Lack of information does not hinder generalized ACGM more than any other primal first-order method while additional information, such as values of strong convexity parameters $\mu_f$ and $\mu_\Psi$ or even an accurate estimate $L_0$ of the curvature around $x_0$, leads to a performance increase at least on par with state-of-the-art of its class.
REFERENCES

[1] A. Beck and M. Teboulle, Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems, IEEE Trans. Image Process., 18 (2009), pp. 2419–2434.
[2] A. Beck and M. Teboulle, A fast iterative shrinkage-thresholding algorithm for linear inverse problems, SIAM J. Imaging Sci., 2 (2009), pp. 183–202.
[3] S. R. Becker, E. J. Candès, and M. C. Grant, Templates for convex cone problems with applications to sparse signal recovery, Math. Program. Comput., 3 (2011), pp. 165–218.
[4] S. Bubeck, Convex Optimization: Algorithms and Complexity, Found. Trends Mach. Learn., 8 (2015), pp. 231–357.
[5] A. Chambolle and C. Dossal, On the convergence of the iterates of the fast iterative shrinkage/thresholding algorithm, J. Optim. Theory Appl., 166 (2015), pp. 968–982.
[6] A. Chambolle and T. Pock, An introduction to continuous optimization for imaging, Acta Numer., 25 (2016), pp. 161–319.
[7] M. I. Florea and S. A. Vorobyov, An accelerated composite gradient method for large-scale composite objective problems, arXiv preprint, arXiv:1612.02352 (2016)
[8] M. I. Florea and S. A. Vorobyov, A robust FISTA-like algorithm, in Proc. of IEEE Intern. Conf. on Acoust., Speech and Signal Process. (ICASSP), New Orleans, USA, 2017, pp. 4521–4525.
[9] T. Hastie, R. Tibshirani, and M. Wainwright, Statistical Learning with Sparsity: The Lasso and Generalizations, CRC Press (2015)
[10] J. L. Hennessy and D. A. Patterson, Computer Architecture, Fifth Edition: A Quantitative Approach, Morgan Kaufmann Publishers Inc., San Francisco, CA, 2011.
[11] L. Lessard, B. Recht, and A. Packard, Analysis and Design of Optimization Algorithms via Integral Quadratic Constraints, SIAM J. Optim., 26 (2016), pp. 57–95.
[12] A. Nemirovsky and D. Yudin, Problem complexity and method efficiency in optimization, John Wiley & Sons, New York, 1983.
[13] Y. Nesterov, A method for unconstrained convex minimization problem with the rate of convergence $O(1/k^2)$, Dokl. AN SSSR (translated as Soviet Math. Docl.), 269 (1983), pp. 543–547.
[14] Y. Nesterov, Introductory Lectures on Convex Optimization. Applied Optimization, vol. 87, Kluwer Academic Publishers, Boston, 2004.
[15] Y. Nesterov, Gradient methods for minimizing composite objective function, Math. Program., Ser. B, 140 (2013), pp. 125–161.
[16] Y. Nesterov, Subgradient methods for huge-scale optimization problems, Math. Program., 146 (2014), pp. 275–297.
[17] Y. Nesterov and S. Stich, Efficiency of the Accelerated Coordinate Descent Method on Structured Optimization Problems, SIAM J. Optim., 27 (2017), pp. 110–123.
[18] N. Parikh, S. P. Boyd, et al., Proximal algorithms, Found. Trends Optim., 1 (2014), pp. 127–239.
[19] R. T. Rockafellar, Convex Analysis, Princeton University Press, Princeton, NJ, 1970.
[20] K. Scheinberg, D. Goldfarb, and X. Bai, Fast first-order methods for composite convex optimization with backtracking, Found. Comput. Math., 14 (2014), pp. 389–417.
[21] K. Slavakis, G. B. Giannakis, and G. Mateos, Modeling and Optimization for Big Data Analytics, IEEE Signal Process. Mag., 31 (2014), pp. 18–31.
[22] W. Su, S. Boyd, and E. J. Candès, A differential equation for modeling Nesterov’s accelerated gradient method: Theory and insights, J. Mach. Learn. Res., 17 (2016), pp. 1–43.
[23] R. Tibshirani, Regression shrinkage and selection via the lasso, J. R. Stat. Soc. B. Methodol., 58 (1996), pp. 267–288.
[24] H. Zou and T. Hastie, Regularization and variable selection via the elastic net, J. R. Stat. Soc. Ser. B. Stat. Methodol., 67 (2005), pp. 301–320.