Florea, Mihai I.; Vorobyov, Sergiy A.

An Accelerated Composite Gradient Method for Large-scale Composite Objective Problems

Published in:
IEEE Transactions on Signal Processing

DOI:
10.1109/TSP.2018.2866409

Published: 15/01/2019

Document Version
Publisher's PDF, also known as Version of record

Please cite the original version:
Florea, M. I., & Vorobyov, S. A. (2019). An Accelerated Composite Gradient Method for Large-scale Composite Objective Problems. *IEEE Transactions on Signal Processing*, *67*(2), 444 - 459. Article 8443140. https://doi.org/10.1109/TSP.2018.2866409

An Accelerated Composite Gradient Method for Large-Scale Composite Objective Problems

Mihai I. Florea, *Student Member, IEEE*, and Sergiy A. Vorobyov, *Fellow, IEEE*

***Abstract*—Various signal processing applications can be expressed as large-scale optimization problems with a composite objective structure, where the Lipschitz constant of the smooth part gradient is either not known, or its local values may only be a fraction of the global value. The smooth part may be strongly convex as well. The algorithms capable of addressing this problem class in its entirety are black-box accelerated first-order methods, related to either Nesterov's Fast Gradient Method or the Accelerated Multistep Gradient Scheme, which were developed and analyzed using the estimate sequence mathematical framework. In this paper, we develop the augmented estimate sequence framework, a relaxation of the estimate sequence. When the lower bounds incorporated in the augmented estimate functions are hyperplanes or parabolae, this framework generates a conceptually simple gap sequence. We use this gap sequence to construct the Accelerated Composite Gradient Method (ACGM), a versatile first-order scheme applicable to any composite problem. Moreover, ACGM is endowed with an efficient dynamic Lipschitz constant estimation (line-search) procedure. We also introduce the wall-clock time unit (WTU), a complexity measure applicable to all first-order methods that accounts for variations in per-iteration complexity and more consistently reflects the running time in practical applications. When analyzed using WTU, ACGM has the best provable convergence rate on the composite problem class, both in the strongly and non-strongly convex cases. Our simulation results confirm the theoretical findings and show the superior performance of our new method.**

***Index Terms*—Acceleration, composite objective, estimate sequence, first-order method, large-scale optimization, line-search, optimization algorithm.**

I. Introduction

NUMEROUS signal processing applications in compressive sensing, medical imaging, geophysics, bioinformatics, and many other areas are currently empowered by large-scale optimization methods (see [1]–[3] and references therein). These applications, due to their size, can only be modeled as optimization problems for which simple operations such as the first-order derivative of the objective function are computationally tractable but complex operations such as Hessian inversion are not (large-scale problems [4]). When these problems are additionally convex, algorithms employing calls to first-order operations (first-order methods) are able to obtain arbitrarily precise estimates of the optimal value given a sufficient number of iterations. Nesterov has demonstrated that first-order methods can be accelerated, when he proposed his breakthrough Fast Gradient Method (FGM) [5]. FGM was constructed using the simple mathematical machinery of the *estimate sequence* [6]. The estimate sequence is a collection of estimate functions, each being a scaled version of a function that incorporates a *global lower bound* while having an optimal value that is a *local upper bound* on the objective function. The local upper bounds tighten as the algorithm progresses, thereby ensuring a provable convergence rate.

Using the estimate sequence, the design process of FGM is straightforward and, by exploiting the structure of smooth problems, simultaneously produces state-of-the art convergence guarantees. FGM converges for non-strongly convex objectives at an optimal rate $\mathcal{O}(1/k^2)$ and for strongly convex objectives at a near-optimal rate $\mathcal{O}((1-\sqrt{q})^{-k})$, where k is the iteration index and q is the inverse condition number of the objective [6]. However, FGM requires that the objective be continuously differentiable with Lipschitz gradient, the Lipschitz constant be known in advance, and the problem be unconstrained.

A broad range of problems, including the most common constrained smooth optimization problems, many inverse problems [7], and several classification and reconstruction problems in imaging [8], have a composite structure, wherein the objective is the sum of a smooth function f with Lipschitz gradient (Lipschitz constant L_f) and a simple function Ψ, that may embed constraints by including the indicator function of the feasible set. By simple function, we mean here that the proximal operator of Ψ is *exact* (for treatment of inexact oracles see, e.g., [9]) and has a negligible cost compared to other operations. We stress that while many specialized methods have been introduced to tackle composite problems with additional structure, such as sparsity (e.g., [10]–[13]), we focus on methods applicable *to the entire problem class*. In particular, we follow the black-box oracle model [14]. Namely, we assume that the exact nature of the objective function is not known by the optimization algorithms (outside the assumptions of the problem class) and they can only obtain information on the problem by calling *oracle functions*. Apart from generality and theoretical simplicity, this model is also well suited for software libraries. Optimization algorithms can be implemented as methods that take as arguments callback oracle functions. Solving a particular problem reduces to providing an implementation of the oracle functions.

To address the demand for fast algorithms applicable to this problem class, as well as to alleviate the need to know L_f in advance, Nesterov has introduced the Accelerated Multistep

Manuscript received November 8, 2017; revised May 7, 2018; accepted May 31, 2018. Date of publication August 21, 2018; date of current version December 10, 2018. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Gesualdo Scutari. This work was partially supported by the Academy of Finland under Grant 299243. This paper was presented in part at the 42nd IEEE International Conference on Acoustics, Speech, and Signal Processing, New Orleans, LA, USA, March 2017. *(Corresponding author: Mihai I. Florea.)*

The authors are with the Department of Signal Processing and Acoustics, Aalto University, Espoo, Finland (e-mail: mihai.florea@aalto.fi; sergiy.vorobyov@aalto.fi).

Digital Object Identifier 10.1109/TSP.2018.2866409

Gradient Scheme (AMGS) [15] that relies on *composite gradients* to overcome the limitations of FGM. This algorithm also adjusts an estimate of L_f at every step (a process often called "line-search" in the literature [7], [16]) that reflects the local curvature of the function. The information collected by AMGS to estimate L_f is reused to advance the algorithm. However, AMGS requires line-search to complete before proceeding to the next iteration. This increases the per-iteration complexity of AMGS to at least twice that of FGM. Consequently, the theoretical convergence guarantees of AMGS, while being better than FGM when measured in iterations, are in fact considerably inferior to FGM in terms of running time (see Appendix A for a detailed analysis).

The Fast Iterative Shrinkage-Thresholding Algorithm (FISTA) [7] decouples the advancement phase from the adjustment phase, stalling the former phase only during backtracks. Decoupling renders the computational complexity of a FISTA iteration comparable to that of FGM. However, FISTA has a fixed $\mathcal{O}(1/k^2)$ provable convergence rate even when the objective is strongly convex, and the line-search strategy cannot decrease the L_f estimate. Similar algorithms to FISTA have been collectively analyzed in [17], but none overcome these drawbacks.

While preparing this manuscript, we became aware of a strongly convex generalization of FISTA, recently introduced in [8], which we designate by FISTA-Chambolle-Pock (FISTA-CP). It has the same convergence guarantees as FGM in both the non-strongly and the strongly convex cases. The monograph [8] hints at but does not explicitly state any line-search strategy. Two recent works also seek to overcome the drawbacks of backtracking FISTA in the strongly convex case.

The first work [18] introduces a family of methods with two notable members. One is the Monteiro-Ortiz-Svaiter (MOS) method, which can be regarded as a simplification of Nesterov's AMGS, obtained by discarding the line-search procedure. MOS has better convergence guarantees than AMGS but it cannot surpass FISTA-CP. The other member is the Adaptive Accelerated (AA) method, which is obtained from MOS by adding an estimate sequence based acceleration heuristic that increases empirical performance on the applications studied in [18] but weakens the theoretical convergence guarantees, making them poorer than those of AMGS (see also Appendix A). The two restart heuristics proposed in [18] are altogether incompatible with the convergence analysis.

The second work [19] proposes a strongly convex Accelerated Proximal Gradient (scAPG) method, which can be regarded as a line-search extension of FISTA-CP applicable to problems where the smooth part f is strongly convex. The convergence guarantees however do not apply outside this scenario.

Thus, a multitude of methods have already been proposed to tackle composite problems with specific additional structure, but none of them successfully combine the strengths of FGM, AMGS, and FISTA.

A. Contributions

- In this work, we give a new interpretation of Nesterov's first-order accelerated optimization algorithms and formulate a generic design pattern for these algorithms based on *local upper bounds* and *global lower bounds*. The global lower bounds are incorporated in the estimate functions whereas the local upper bounds are defined separately.
- Nesterov's estimate sequence can be relaxed to produce an *augmented estimate sequence*. Augmentation renders the estimate sequence invariant to the tightness of the global lower bounds.
- When these lower bounds take the form of generalized parabolae (hyperplanes or quadratic functions with Hessians equal to multiples of the identity matrix), the augmented estimate sequence property can be insured by maintaining a non-increasing (Lyapunov property) *gap sequence*.
- We provide, using the above design pattern and the gap sequence, a step-by-step derivation of our Accelerated Composite Gradient Method (ACGM), a versatile first-order scheme for the class of large-scale problems with composite objectives, which has the convergence guarantees of FGM in both the non-strongly and strongly convex cases. ACGM is equipped with an efficient adaptive line-search procedure that is decoupled from the advancement phase at every iteration. ACGM does not require a priori knowledge of the Lipschitz constant and can converge even when the Lipschitz property holds only locally.
- ACGM is derived in an estimate sequence based form but it can be brought to an equivalent extrapolation based form that is more similar to FISTA and its extensions.
- We introduce the wall-clock time unit (WTU), a complexity measure that accounts for variations in the per-iteration complexity of black-box optimization algorithms. WTU more accurately reflects the actual performance of such algorithms in practical applications.
- When analyzed using WTU, ACGM has the best provable convergence rate both in the strongly and non-strongly convex cases.
- We corroborate the theoretical arguments with simulation results. Specifically, we show that on a popular instance of the non-strongly convex l_1-regularized image deblurring problem and on a random instance of the strongly convex logistic regression with elastic net regularization problem, each with the Lipschitz constant assumed unknown, our method surpasses the state-of-the-art in terms of WTU usage.

B. Assumptions and Notation

We consider the following convex optimization problem

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} F(\boldsymbol{x}) \triangleq f(\boldsymbol{x}) + \Psi(\boldsymbol{x}),$$

where $\boldsymbol{x}$ is a vector of n optimization variables. In this work, we consider only *large-scale* problems [4]. The composite objective F has a non-empty set of optimal points X^*. Function $f:\mathbb{R}^n\to\mathbb{R}$ is convex differentiable on $\mathbb{R}^n$ with Lipschitz gradient (Lipschitz constant $L_f>0$) and a strong convexity parameter $\mu_f\geq 0$. The regularizer $\Psi:\mathbb{R}^n\to\mathbb{R}\cup\{\infty\}$ is a proper lower semicontinuous convex function with a strong convexity parameter μ_Ψ. This implies that F has a strong convexity parameter $\mu=\mu_f+\mu_\Psi$. The regularizer Ψ embeds constraints by being infinite outside the feasible set. It does not have to be differentiable. However, its proximal map, given by

$$\operatorname{prox}_{\tau\Psi}(\boldsymbol{x}) \triangleq \operatorname*{arg\,min}_{\boldsymbol{z}\in\mathbb{R}^n}\left(\Psi(\boldsymbol{z})+\frac{1}{2\tau}\|\boldsymbol{z}-\boldsymbol{x}\|_2^2\right),$$

for all $\boldsymbol{x} \in \mathbb{R}^n$ and $\tau > 0$ can be computed with complexity $\mathcal{O}(n)$. Here $\|.\|_2$ denotes the Euclidean norm. The optimization problem is treated by algorithms in a *black-box* setting [14], i.e. algorithms can only access oracle functions $f(\boldsymbol{x})$, $\nabla f(\boldsymbol{x})$, $\Psi(\boldsymbol{x})$, and $\text{prox}_{\tau\Psi}(\boldsymbol{x})$, with arguments $\boldsymbol{x} \in \mathbb{R}^n$ and $\tau > 0$.

We define a parabola as a quadratic function $\psi : \mathbb{R}^n \to \mathbb{R}$ of the form

$$\psi(\boldsymbol{x}) \triangleq \psi^* + \frac{\gamma}{2}\|\boldsymbol{x} - \boldsymbol{v}\|_2^2, \quad \boldsymbol{x} \in \mathbb{R}^n,$$

where $\gamma > 0$ gives the curvature, $\boldsymbol{v} \in \mathbb{R}^n$ is the vertex, and ψ^* is the optimal value. We also define $\mathcal{P}$ as the set of all parabolae, $\mathcal{H}$ as the set of all linear functions $h : \mathbb{R}^n \to \mathbb{R}$ (which we denote as hyperplanes), and $\mathcal{G}$ as the set of generalized parabolae, $\mathcal{G} \triangleq \mathcal{P} \cup \mathcal{H}$. We define two abbreviated expressions, $P_{f,\boldsymbol{y}}(\boldsymbol{x}) \in \mathcal{H}$ and $Q_{f,\gamma,\boldsymbol{y}}(\boldsymbol{x}) \in \mathcal{G}$, as

$$\begin{aligned} P_{f,\boldsymbol{y}}(\boldsymbol{x}) &\triangleq f(\boldsymbol{y}) + \langle \nabla f(\boldsymbol{y}), \boldsymbol{x} - \boldsymbol{y} \rangle, \\ Q_{f,\gamma,\boldsymbol{y}}(\boldsymbol{x}) &\triangleq P_{f,\boldsymbol{y}}(\boldsymbol{x}) + \frac{\gamma}{2}\|\boldsymbol{x} - \boldsymbol{y}\|_2^2, \end{aligned} \tag{1}$$

for any $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ and $\gamma > 0$, where $\langle ., . \rangle$ denotes the inner product. Using expression Q, we introduce the proximal gradient operator $T_{f,\Psi,L}(\boldsymbol{y})$ as

$$\begin{aligned} T_{f,\Psi,L}(\boldsymbol{y}) &\triangleq \underset{\boldsymbol{x} \in \mathbb{R}^n}{\arg\min} \left(Q_{f,L,\boldsymbol{y}}(\boldsymbol{x}) + \Psi(\boldsymbol{x}) \right) \\ &= \text{prox}_{\frac{1}{L}\Psi}\left(\boldsymbol{y} - \frac{1}{L}\nabla f(\boldsymbol{y}) \right), \quad \boldsymbol{y} \in \mathbb{R}^n, \end{aligned} \tag{2}$$

where $L > 0$ is a parameter corresponding to the inverse of the step size.

II. Theoretical Building Blocks

First, we present the mathematical machinery used in constructing ACGM. We begin this section with a novel interpretation of Nesterov's estimate sequence, we proceed by introducing a generic design pattern for estimate sequence based algorithms, and conclude with the properties of the composite gradient that allow us to design the relaxed lower bounds of ACGM.

A. Estimate Sequence

For the class of composite problems with non-strongly convex objectives, regardless of the optimization algorithm used, the convergence of the iterates can be arbitrarily slow [6], [20]. Consequently, we express the convergence rate of first-order schemes on the entire composite problem class as the decrease rate of the distance between the objective value and the optimal value. We define a convergence guarantee (provable convergence rate) as the decrease rate of a theoretical upper bound on this distance. When designing algorithms, we index objective values based on iterations.[1] The bound is expressed in terms of points in the domain space (see also [15]) as

$$A_k(F(\boldsymbol{x}_k) - F(\boldsymbol{x}^*)) \leq \frac{1}{2}\|\boldsymbol{x}_0 - \boldsymbol{x}^*\|_2^2, \tag{3}$$

for any $\boldsymbol{x}^* \in X^*$ and $k \geq 0$. Without loss of generality, we will fix $\boldsymbol{x}^*$ to be an arbitrary element of X^* throughout the remainder of this work. The weight sequence $\{A_k\}_{k\geq 0}$ with $A_k > 0$ for all $k \geq 1$ gives the convergence guarantees. Since the starting point $\boldsymbol{x}_0$ is assumed to be arbitrary, the composite function value $F(\boldsymbol{x}_0)$ may not be finite and no guarantee can be given for $k = 0$. Therefore, A_0 is set to 0 to ensure that (3) holds.

The provable convergence rate expression (3) translates to

$$A_k F(\boldsymbol{x}_k) \leq H_k, \tag{4}$$

where

$$H_k \triangleq A_k F(\boldsymbol{x}^*) + \frac{1}{2}\|\boldsymbol{x}_0 - \boldsymbol{x}^*\|_2^2, \quad k \geq 0, \tag{5}$$

is the highest allowable upper bound on the weighted objective values $A_k F(\boldsymbol{x}_k)$. The convexity of F ensures that there exists a sequence $\{W_k\}_{k\geq 1}$ of global convex lower bounds on F, namely

$$F(\boldsymbol{x}) \geq W_k(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^n, \quad k \geq 1. \tag{6}$$

We define an estimate sequence $\{\psi_k(\boldsymbol{x})\}_{k\geq 0}$ as

$$\psi_k(\boldsymbol{x}) \triangleq A_k W_k(\boldsymbol{x}) + \frac{\gamma_0}{2}\|\boldsymbol{x} - \boldsymbol{x}_0\|_2^2, \ 0 < \gamma_0 \leq 1, \ k \geq 0. \tag{7}$$

Here ψ_k for $k \geq 0$ are estimate functions and γ_0 is the curvature of the initial estimate function ψ_0. Since $A_0 = 0$, there is no need to define W_0. Both AMGS and FGM are built to maintain the following estimate sequence property[2]

$$A_k F(\boldsymbol{x}_k) \leq \psi_k^*, \tag{8}$$

where

$$\psi_k^* \triangleq \min_{\boldsymbol{x} \in \mathbb{R}^n} \psi_k(\boldsymbol{x}), \quad k \geq 0.$$

The estimate sequence property states that the estimate function optimal value is a *scaled* (by A_k) *local* (at $\boldsymbol{x}_k$) *upper bound* on the objective F. Since the weights are increasing, it follows that the local upper bounds $\frac{1}{A_k}\psi_k^*$ for $k \geq 1$ are increasingly tight, while incorporating the global lower bounds W_k. The provable convergence rate bound in (4) follows naturally from (6), (7), and (8). Thus, we have

$$A_k F(\boldsymbol{x}_k) \leq \psi_k^* \leq \psi_k(\boldsymbol{x}^*) \leq H_k, \quad k \geq 0.$$

The estimate sequence property in (8) is more stringent than the provable convergence rate expression in (4). The gap between ψ_k^* and H_k is large and, as we shall see in Subsection III-B, can be reduced to yield a relaxation of the estimate sequence with remarkable properties.

B. A Design Pattern for Nesterov's First-order Accelerated Algorithms

Nesterov's FGM and AMGS share the structure outlined in Algorithm 1.

Algorithm 1 takes as input the starting point $\boldsymbol{x}_0 \in \mathbb{R}^n$, an initial estimate of the Lipschitz constant $L_0 > 0$, the total number of iterations $K > 0$, the initial weight $A_0 \geq 0$, and the initial curvature $0 < \gamma_0 \leq 1$. At every iteration k, the future value of the main iterate $\boldsymbol{x}_{k+1}$ is generated using majorization minimization, i.e., it is set as the minimum of u_{k+1}, a local upper

[1] This does not necessarily reflect the actual performance of the algorithm. See Section IV for a detailed discussion.

[2] The definition in (7) corresponds to the "newer variant", introduced in [15] to analyze AMGS in the context of composite functions and, in particular, of infeasible start. For FGM, the estimate sequence definition differs slightly (see [6], [9]).

TABLE I
DESIGN CHOICES OF FGM AND AMGS AT EVERY ITERATION $k \geq 0$

Symbol	In FGM	In AMGS
$w_{k+1}(\boldsymbol{x})$	$Q_{f,\mu,\boldsymbol{y}_{k+1}}(\boldsymbol{x})$	$P_{f,\boldsymbol{x}_{k+1}}(\boldsymbol{x}) + \Psi(\boldsymbol{x})$
$u_{k+1}(\boldsymbol{x})$	$Q_{f,L_f,\boldsymbol{y}_{k+1}}(\boldsymbol{x})$	$Q_{f,L_{k+1},\boldsymbol{y}_{k+1}}(\boldsymbol{x}) + \Psi(\boldsymbol{x})$
$\mathcal{F}_a(\psi_k, A_k, L_{k+1})$	Solution $a > 0$ of $L_f a^2 = (A_k + a)(\gamma_k + \mu a)$	Solution $a > 0$ of $L_{k+1} a^2 = 2(A_k + a)(1 + \mu A_k)$
$\mathcal{F}_{\boldsymbol{y}}(\boldsymbol{x}_k, \psi_k, A_k, a_{k+1})$	$\frac{A_k \gamma_{k+1} \boldsymbol{x}_k + a_{k+1} \gamma_k \boldsymbol{v}_k}{A_k \gamma_{k+1} + a_{k+1} \gamma_k}$	$\frac{A_k \boldsymbol{x}_k + a_{k+1} \boldsymbol{v}_k}{A_k + a_{k+1}}$

Algorithm 1: A Design Pattern for Nesterov's First-order Accelerated Algorithms.

1: $\psi_0(\boldsymbol{x}) = A_0 F(\boldsymbol{x}_0) + \frac{\gamma_0}{2}\|\boldsymbol{x} - \boldsymbol{x}_0\|_2^2$
2: **for** $k = 0, \ldots, K-1$ **do**
3: $\quad L_{k+1} = \mathcal{S}(\boldsymbol{x}_k, \psi_k, A_k, L_k)$ "line-search"
4: $\quad a_{k+1} = \mathcal{F}_a(\psi_k, A_k, L_{k+1})$
5: $\quad \boldsymbol{y}_{k+1} = \mathcal{F}_y(\boldsymbol{x}_k, \psi_k, A_k, a_{k+1})$
6: $\quad A_{k+1} = A_k + a_{k+1}$
7: $\quad \boldsymbol{x}_{k+1} = \arg\min_{\boldsymbol{x} \in \mathbb{R}^n} u_{k+1}(\boldsymbol{x})$
8: $\quad \psi_{k+1}(\boldsymbol{x}) = \psi_k(\boldsymbol{x}) + a_{k+1} w_{k+1}(\boldsymbol{x})$
9: **end for**

bound on F (Algorithm 1, line 7). Note that u_{k+1} is not related to ψ_k^*. The estimate function ψ_k is incremented with a global lower bound w_{k+1} weighted by a_{k+1} (Algorithm 1, line 8). This ensures that the next estimate function ψ_{k+1} retains the canonical form in (7), where the lower bounds W_k are given by

$$w_{k+1} = \frac{1}{A_k} \sum_{i=1}^{k} a_i w_i(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^n, \quad k \geq 1.$$

The weight a_{k+1} and the test point $\boldsymbol{y}_{k+1}$ are obtained as functions $\mathcal{F}_a$ and $\mathcal{F}_y$, respectively, of the state variables at each iteration (Algorithm 1, lines 4 and 5). These functions are derived in the algorithm design stage to guarantee that the estimate sequence property in (8) carries over to the next iterate, regardless of the algorithmic state. The line-search procedure $\mathcal{S}$ (Algorithm 1, line 3) outputs an estimate of L_f, denoted by L_{k+1}.

Table I lists the expressions of functions $\mathcal{F}_a$ and $\mathcal{F}_y$ as well as the lower bounds w_{k+1} and upper bounds u_{k+1} for both FGM and AMGS. Note that FGM does not use line-search nor the input parameter L_0. It assumes that $\Psi(\boldsymbol{x}) = 0$ and that L_f is known in advance. It defines the local upper bounds based directly on L_f. The estimate functions of FGM and AMGS take the form of

$$\psi_k^{\mathrm{FGM}}(\boldsymbol{x}) = (\psi_k^*)^{\mathrm{FGM}} + \frac{\gamma_k}{2}\|\boldsymbol{x} - \boldsymbol{v}_k\|_2^2,$$

$$\psi_k^{\mathrm{AMGS}}(\boldsymbol{x}) = (\psi_k^*)^{\mathrm{AMGS}} + \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{v}_k\|_2^2 + A_k \Psi(\boldsymbol{x}),$$

for all $\boldsymbol{x} \in \mathbb{R}^n$ and $k \geq 0$. Both methods enforce $\gamma_0 = 1$ (our notation differs from the one in [6]). The convergence analysis of AMGS requires that $A_0 = 0$ (also argued in Subsection II-A) while for FGM we have $0 < A_0 \leq 1/L_f$.

Under the above assumptions, by replacing the symbols in Algorithm 1 with the corresponding expressions in Table I, we recover FGM and AMGS, respectively.

C. Composite Gradient

A further link between FGM and AMGS has been provided in [15] by means of the *composite gradient*, defined as

$$g_L(\boldsymbol{y}) \triangleq L\,(\boldsymbol{y} - T_{f,\Psi,L}(\boldsymbol{y})), \quad \boldsymbol{y} \in \mathbb{R}^n, \quad L > 0. \tag{9}$$

As we shall see in (18), there is no need specify functional parameters. The composite gradient substitutes the gradient for composite functions and shares many of its properties. Most notably, the descent update (Algorithm 1, line 7) in FGM, given by

$$\boldsymbol{x}_{k+1} = \boldsymbol{y}_{k+1} - \frac{1}{L_f}\nabla f(\boldsymbol{y}_{k+1}),$$

can be written similarly in AMGS using the composite gradient as

$$\boldsymbol{x}_{k+1} = \boldsymbol{y}_{k+1} - \frac{1}{L_{k+1}} g_{L_{k+1}}(\boldsymbol{y}_{k+1}).$$

In addition, the descent rule [6], which for FGM takes the form of

$$f(\boldsymbol{x}_{k+1}) \leq f(\boldsymbol{y}_{k+1}) - \frac{1}{2L_f}\|\nabla f(\boldsymbol{y}_{k+1})\|_2^2, \tag{10}$$

is obeyed by the composite gradient in AMGS as well (see Lemma 1), that is,

$$F(\boldsymbol{x}_{k+1}) \leq F(\boldsymbol{y}_{k+1}) - \frac{1}{2L_{k+1}}\|g_{L_{k+1}}(\boldsymbol{y}_{k+1})\|_2^2.$$

These properties suggest that FGM could be applied to composite objectives simply by replacing the gradient call with a composite gradient call, yielding an algorithm that has the superior convergence guarantees of FGM and the applicability of AMGS.

III. ACGM

The convergence analysis of FGM in [6] requires only two properties of the gradient to hold: the descent rule in (10) and the supporting generalized parabola condition, i.e., $Q_{f,\mu,\boldsymbol{y}_{k+1}}$ is a lower bound on function f for all $k \geq 0$. However, the naive extension of $Q_{f,\mu,\boldsymbol{y}_{k+1}}(\boldsymbol{x})$ to composite gradients, written as

$$F(\boldsymbol{y}_{k+1}) + \langle g_{L_{k+1}}(\boldsymbol{y}_{k+1}), \boldsymbol{x} - \boldsymbol{y}_{k+1}\rangle + \frac{\mu}{2}\|\boldsymbol{x} - \boldsymbol{y}_{k+1}\|_2^2, \tag{11}$$

$\boldsymbol{x} \in \mathbb{R}^n$, is *not guaranteed* to be a valid lower bound on F for *any* value of $L_{k+1} > 0$. Hence, this convergence analysis of FGM does not apply to composite objectives.

A. Relaxed Lower Bound

We seek a suitable replacement for the FGM supporting generalized parabolae, bearing in mind that the accuracy of the lower bounds at every iteration impacts the convergence rate of the algorithm. At every iteration k, the lower bound in FGM takes the form of an approximate second order Taylor expansion of f at $\boldsymbol{y}_{k+1}$. For ACGM, we produce a similar lower bound on F by transferring all strong convexity, if any, from Ψ to f as

$$f'(\boldsymbol{x}) \triangleq f(\boldsymbol{x}) + \frac{\mu_\Psi}{2}\|\boldsymbol{x}-\boldsymbol{x}_0\|_2^2, \tag{12}$$

$$\Psi'(\boldsymbol{x}) \triangleq \Psi(\boldsymbol{x}) - \frac{\mu_\Psi}{2}\|\boldsymbol{x}-\boldsymbol{x}_0\|_2^2, \tag{13}$$

$\boldsymbol{x} \in \mathbb{R}^n$. Note that the center of strong convexity in (12) and (13) can be any point in $\mathbb{R}^n$. We choose $\boldsymbol{x}_0$ only for convenience. Function f' has Lipschitz gradient with constant $L_{f'} = L_f + \mu_\Psi$ and a strong convexity parameter $\mu_{f'} = \mu$. Naturally, this transfer does not alter the objective function

$$F(\boldsymbol{x}) = f(\boldsymbol{x}) + \Psi(\boldsymbol{x}) = f'(\boldsymbol{x}) + \Psi'(\boldsymbol{x})$$

and gives rise to the following remarkable property.

Proposition 1: By transferring convexity as in (12) we have

$$Q_{f',L+\mu_\Psi,\boldsymbol{y}}(\boldsymbol{x}) = Q_{f,L,\boldsymbol{y}}(\boldsymbol{x}) + \frac{\mu_\Psi}{2}\|\boldsymbol{x}-\boldsymbol{x_0}\|_2^2,$$

for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ and $L > 0$.

Proof: See Appendix B. ■

From Proposition 1 and (12) it follows that the descent condition for f at every iteration k, given by

$$f(\boldsymbol{x}_{k+1}) \le Q_{f,L_{k+1},\boldsymbol{y}_{k+1}}(\boldsymbol{x}_{k+1}), \tag{14}$$

is equivalent to that of f', stated as

$$f'(\boldsymbol{x}_{k+1}) \le Q_{f',L'_{k+1},\boldsymbol{y}_{k+1}}(\boldsymbol{x}_{k+1}), \tag{15}$$

where $L'_{k+1} \triangleq L_{k+1} + \mu_\Psi$.

When designing ACGM, we assume no upper bound on Ψ. Therefore, we have to choose a composite parabolic upper bound on F at every iteration $k \ge 0$, that is,

$$u_{k+1}(\boldsymbol{x}) = Q_{f,L_{k+1},\boldsymbol{y}_{k+1}}(\boldsymbol{x}) + \Psi(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^n. \tag{16}$$

From Proposition 1 we can also see that the strong convexity transfer in (12) and (13) does not alter the upper bound, namely

$$u_{k+1}(\boldsymbol{x}) = Q_{f',L'_{k+1},\boldsymbol{y}_{k+1}}(\boldsymbol{x}) + \Psi'(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^n. \tag{17}$$

The invariance shown in (16) and (17) implies that the update in line 7 of Algorithm 1 remains unchanged as well:

$$\boldsymbol{x}_{k+1} = T_{f,\Psi,L_{k+1}}(\boldsymbol{y}_{k+1}) = T_{f',\Psi',L'_{k+1}}(\boldsymbol{y}_{k+1}). \tag{18}$$

We are now ready to formulate the sought after lower bound. The following result can be regarded as a generalization of Theorem 2.2.7 in [6], Lemma 2.3 in [7], and (4.37) in [8].

Lemma 1: If the descent condition in (14) holds at iteration $k \ge 0$, then the objective F is lower bounded as

$$F(\boldsymbol{x}) \ge \mathcal{R}_{L'_{k+1},\boldsymbol{y}_{k+1}}(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^n,$$

where we denote with $\mathcal{R}_{L'_{k+1},\boldsymbol{y}_{k+1}}(\boldsymbol{x})$ the *relaxed supporting generalized parabola* of F at $\boldsymbol{y}_{k+1}$ using inverse step size L'_{k+1}, given by

$$\mathcal{R}_{L'_{k+1},\boldsymbol{y}_{k+1}}(\boldsymbol{x}) \triangleq F(\boldsymbol{x}_{k+1}) + \frac{1}{2L'_{k+1}}\|g_{L'_{k+1}}(\boldsymbol{y}_{k+1})\|_2^2 + \langle g_{L'_{k+1}}(\boldsymbol{y}_{k+1}), \boldsymbol{x}-\boldsymbol{y}_{k+1}\rangle + \frac{\mu}{2}\|\boldsymbol{x}-\boldsymbol{y}_{k+1}\|_2^2, \quad \boldsymbol{x} \in \mathbb{R}^n,$$

with $\boldsymbol{x}_{k+1}$ given by (18).

Proof: See Appendix C. ■

The relaxed supporting generalized parabola thus differs from the naive extension of $Q_{f,\mu,\boldsymbol{y}_{k+1}}(\boldsymbol{x})$ to composite gradients in (11) by a small constant factor.

B. Augmented Estimate Sequence

Recall that the estimate sequence property in (8) produces a gap between ψ_k^* and H_k. This allows us to introduce the more relaxed *augmented estimate sequence* $\{\psi'_k(\boldsymbol{x})\}_{k\ge 0}$ which we define, using the notation and conventions from Subsection II-A, as

$$\psi'_k(\boldsymbol{x}) \triangleq \psi_k(\boldsymbol{x}) + A_k(F(\boldsymbol{x}^*) - W_k(\boldsymbol{x}^*)), \quad \boldsymbol{x} \in \mathbb{R}^n, \quad k \ge 0. \tag{19}$$

Augmentation consists only of adding a non-negative constant (due to the lower bound property of W_k) to the estimate function, thus preserving its curvature and vertex. The augmented estimate sequence property, given as

$$A_k F(\boldsymbol{x}_k) \le \psi_k'^*, \quad k \ge 0, \tag{20}$$

can be used to derive the provable convergence rate because, along with definitions (5), (7), and (19), it implies that

$$\begin{aligned} A_k F(\boldsymbol{x}_k) &\le \psi_k'^* = \psi_k^* + A_k(F(\boldsymbol{x}^*) - W_k(\boldsymbol{x}^*)) \\ &\le \psi_k^* + H_k - \psi_k(\boldsymbol{x}^*) \le H_k, \quad k \ge 0. \end{aligned}$$

Note that by subtracting the lower bound constant term $W_k(\boldsymbol{x}^*)$, augmentation renders property (20) invariant to the tightness of the lower bounds.

C. Gap Sequence

Maintaining the augmented estimate sequence property in (20) across iterations is equivalent to ensuring that the gap between the weighted function values and the augmented estimate function optimal value, defined as

$$\Gamma_k \triangleq A_k F(\boldsymbol{x}_k) - \psi_k'^*, \quad k \ge 0,$$

is non-positive. Given that initially $\Gamma_0 = A_0 F(\boldsymbol{x}_0) - \psi_0'^* = 0$, a sufficient condition for this guarantee is that Γ_k is monotonically decreasing, that is

$$\Gamma_{k+1} \le \Gamma_k, \quad k \ge 0. \tag{21}$$

Since the initial estimate function is a parabola and the lower bounds are generalized parabolae, we can write the estimate function at any iteration k, along with its augmented variant, as the following parabolae:

$$\psi_k(\boldsymbol{x}) = \psi_k^* + \frac{\gamma_k}{2}\|\boldsymbol{x}-\boldsymbol{v}_k\|_2^2, \tag{22}$$

$$\psi'_k(\boldsymbol{x}) = \psi_k'^* + \frac{\gamma_k}{2}\|\boldsymbol{x}-\boldsymbol{v}_k\|_2^2, \tag{23}$$

$\boldsymbol{x} \in \mathbb{R}^n$. The gap between $A_k F(\boldsymbol{x}_k)$ and $\psi_k'^*$ can be expressed as

$$\begin{aligned}\Gamma_k &\overset{(19)}{=} A_k(F(\boldsymbol{x}_k) - F(\boldsymbol{x}^*)) + A_k W_k(\boldsymbol{x}^*) - \psi_k^* \\ &\overset{(7)}{=} A_k(F(\boldsymbol{x}_k) - F(\boldsymbol{x}^*)) + \psi_k(\boldsymbol{x}^*) - \psi_k^* - \frac{\gamma_0}{2}\|\boldsymbol{x}^* - \boldsymbol{x}_0\|_2^2 \\ &\overset{(22)}{=} A_k(F(\boldsymbol{x}_k) - F(\boldsymbol{x}^*)) + \frac{\gamma_k}{2}\|\boldsymbol{v}_k - \boldsymbol{x}^*\|_2^2 - \frac{\gamma_0}{2}\|\boldsymbol{x}^* - \boldsymbol{x}_0\|_2^2\end{aligned}$$

for all $k \geq 0$. We define the *gap sequence* $\{\Delta_k\}_{k\geq 0}$ as

$$\Delta_k \triangleq A_k(F(\boldsymbol{x}_k) - F(\boldsymbol{x}^*)) + \frac{\gamma_k}{2}\|\boldsymbol{v}_k - \boldsymbol{x}^*\|_2^2, \quad k \geq 0.$$

With the quantity $\frac{\gamma_0}{2}\|\boldsymbol{x}^* - \boldsymbol{x}_0\|_2^2$ being constant across iterations, the sufficient condition (21) can be rewritten as

$$\Delta_{k+1} \leq \Delta_k, \quad k \geq 0. \tag{24}$$

The benefits of the augmented estimate sequence now become evident. We have replaced the estimate sequence property with a gap sequence that has a simple closed form. The gap sequence is an example of a Lyapunov (non-increasing) function, widely used in the convergence analysis of optimization schemes (e.g., [21]).

D. Formulating ACGM

We proceed with the design of our method, ACGM, based on the pattern presented in Algorithm 1. The building blocks are as follows:

1) The Lyapunov property of the gap sequence in (24);
2) The composite parabolic upper bounds in (16);
3) The relaxed supporting generalized parabola lower bounds from Lemma 1, namely

$$w_{k+1}(\boldsymbol{x}) = \mathcal{R}_{L'_{k+1}, \boldsymbol{y}_{k+1}}(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^n, \quad k \geq 0. \tag{25}$$

The upper bounds in (16) imply that line 7 of Algorithm 1 is the proximal gradient step in (18). For the relaxed supporting generalized parabola to be a valid global lower bound on F, Lemma 1 requires that, at every iteration k, the descent condition for f in (14) holds. This is assured in the worst case when $L_{k+1} \geq L_f$. The structure of the lower bounds implies that the estimate functions and their augmented counterparts take the form in (22) and (23), respectively. Substituting the lower bound from (25) in the estimate sequence update in line 8 of Algorithm 1 and differentiating with respect to $\boldsymbol{x}$ gives the curvature and vertex update rules for all $k \geq 0$ as

$$\gamma_{k+1} = \gamma_k + a_{k+1}\mu, \tag{26}$$

$$\boldsymbol{v}_{k+1} = \frac{1}{\gamma_{k+1}}\left(\gamma_k \boldsymbol{v}_k - a_{k+1}(g_{L'_{k+1}}(\boldsymbol{y}_{k+1}) - \mu \boldsymbol{y}_{k+1})\right). \tag{27}$$

Next, we devise update rules for a_{k+1} and $\boldsymbol{y}_{k+1}$ to ensure that the Lyapunov property of the gap sequence in (24) is satisfied at every iteration $k \geq 0$ for any algorithmic state.

Theorem 1: If at iteration $k \geq 0$, the descent condition for f in (14) holds, then

$$\Delta_{k+1} + \mathcal{A}_{k+1} + \mathcal{B}_{k+1} \leq \Delta_k,$$

where subexpressions $\mathcal{A}_{k+1}$, $\mathcal{B}_{k+1}$, $\boldsymbol{s}_{k+1}$, and Y_{k+1} are, respectively, defined as

$$\begin{aligned}\mathcal{A}_{k+1} &\triangleq \frac{1}{2}\left(\frac{A_{k+1}}{L'_{k+1}} - \frac{a_{k+1}^2}{\gamma_{k+1}}\right)\|g_{L'_{k+1}}(\boldsymbol{y}_{k+1})\|_2^2, \\ \mathcal{B}_{k+1} &\triangleq \frac{1}{\gamma_{k+1}}\left\langle g_{L'_{k+1}}(\boldsymbol{y}_{k+1}) - \frac{\mu}{2Y_{k+1}}\boldsymbol{s}_{k+1}, \boldsymbol{s}_{k+1}\right\rangle, \\ \boldsymbol{s}_{k+1} &\triangleq A_k\gamma_{k+1}\boldsymbol{x}_k + a_{k+1}\gamma_k \boldsymbol{v}_k - Y_{k+1}\boldsymbol{y}_{k+1}, \\ Y_{k+1} &\triangleq A_k\gamma_{k+1} + a_{k+1}\gamma_k.\end{aligned}$$

Proof: See Appendix D. ■

Theorem 1 implies that (24) holds if, regardless of the algorithmic state, $\mathcal{A}_{k+1} \geq 0$ and $\mathcal{B}_{k+1} \geq 0$. The former inequality translates to

$$A_{k+1}\gamma_{k+1} \geq L'_{k+1}a_{k+1}^2 = (L_{k+1} + \mu_\Psi)a_{k+1}^2. \tag{28}$$

The vector terms in $\mathcal{B}_{k+1}$ may form an obtuse angle so we impose $\mathcal{B}_{k+1} = 0$ by setting $\boldsymbol{s}_{k+1} = 0$. This gives an expression for $\mathcal{F}_y$ in Algorithm 1 in the form of

$$\begin{aligned}\boldsymbol{y}_{k+1} &= \mathcal{F}_y(\boldsymbol{x_k}, \psi_k, A_k, a_{k+1}) \\ &= \frac{1}{A_k\gamma_{k+1} + a_{k+1}\gamma_k}\left(A_k\gamma_{k+1}\boldsymbol{x}_k + a_{k+1}\gamma_k \boldsymbol{v}_k\right),\end{aligned} \tag{29}$$

where γ_{k+1} is obtained from (26).

We choose the most aggressive accumulated weight update by enforcing equality in (28) and by setting $\gamma_0 = 1$, which ensures that γ_{k+1} is as large as possible. Update (28) becomes

$$(L_{k+1} + \mu_\Psi)a_{k+1}^2 = A_{k+1}\gamma_{k+1} \overset{(26)}{=} (A_k + a_{k+1})(\gamma_k + \mu a_{k+1}). \tag{30}$$

Given that $a_{k+1}, L_{k+1} > 0$ and $A_k \geq 0$, we can write $\mathcal{F}_a$ in closed form as

$$\begin{aligned}a_{k+1} = \mathcal{F}_a(\psi_k, A_k, L_{k+1}) &= \frac{1}{2(L_{k+1} - \mu_f)} \\ &\left(\gamma_k + A_k\mu + \sqrt{(\gamma_k + A_k\mu)^2 + 4(L_{k+1} - \mu_f)A_k\gamma_k}\right).\end{aligned} \tag{31}$$

By using the definition of the composite gradient in (9), the update rule for the vertices in (27) becomes

$$\begin{aligned}\boldsymbol{v}_{k+1} &= \frac{1}{\gamma_{k+1}}\left(\gamma_k \boldsymbol{v}_k - a_{k+1}(L'_{k+1}(\boldsymbol{y}_{k+1} - \boldsymbol{x}_{k+1}) - \mu \boldsymbol{y}_{k+1})\right) \\ &= \frac{1}{\gamma_{k+1}}(\gamma_k \boldsymbol{v}_k + a_{k+1}(L_{k+1} + \mu_\Psi)\boldsymbol{x}_{k+1} \\ &\quad - a_{k+1}(L_{k+1} - \mu_f)\boldsymbol{y}_{k+1}).\end{aligned} \tag{32}$$

Finally, we select the same Armijo-type [22] line-search strategy $\mathcal{S}_A$ as AMGS [15], with parameters $r_u > 1$ and $0 < r_d \leq 1$ as the increase and decrease rates, respectively, of the Lipschitz constant estimate.

In summary, we have established the values of the initial parameters ($A_0 = 0$, $\gamma_0 = 1$, and $\boldsymbol{v}_0 = \boldsymbol{x}_0$), the upper bounds in (16) which give the iterate update in (18), the relaxed supporting generalized parabola lower bounds in (25) that yield the curvature update in (26) and the vertex update in (32), the line-search strategy $\mathcal{S}_A$, as well as the expressions of functions $\mathcal{F}_a$ in (31) and $\mathcal{F}_y$ in (29). Based on Algorithm 1, we can now write down ACGM as listed in Algorithm 2. Temporary estimates of

Algorithm 2: ACGM in Estimate Sequence Form
ACGM$(\boldsymbol{x_0}, L_0, \mu_f, \mu_\Psi, K)$.

$\boldsymbol{v}_0 = \boldsymbol{x}_0,\ \mu = \mu_f + \mu_\Psi,\ A_0 = 0,\ \gamma_0 = 1$
for $k = 0, \ldots, K-1$ **do**
  $\hat{L}_{k+1} := r_d L_k$
  loop
    $\hat{a}_{k+1} := \frac{1}{2(\hat{L}_{k+1} - \mu_f)} \left(\gamma_k + A_k\mu + \sqrt{(\gamma_k + A_k\mu)^2 + 4(\hat{L}_{k+1} - \mu_f)A_k\gamma_k}\right)$
    $\hat{A}_{k+1} := A_k + \hat{a}_{k+1}$
    $\hat{\gamma}_{k+1} := \gamma_k + \hat{a}_{k+1}\mu$
    $\hat{\boldsymbol{y}}_{k+1} := \frac{1}{A_k\hat{\gamma}_{k+1} + \hat{a}_{k+1}\gamma_k} (A_k\hat{\gamma}_{k+1}\boldsymbol{x}_k + \hat{a}_{k+1}\gamma_k\boldsymbol{v}_k)$
    $\hat{\boldsymbol{x}}_{k+1} := \operatorname{prox}_{\frac{1}{\hat{L}_{k+1}}\Psi}\left(\hat{\boldsymbol{y}}_{k+1} - \frac{1}{\hat{L}_{k+1}}\nabla f(\hat{\boldsymbol{y}}_{k+1})\right)$
    if $f(\hat{\boldsymbol{x}}_{k+1}) \le Q_{f,\hat{L}_{k+1},\hat{\boldsymbol{y}}_{k+1}}(\hat{\boldsymbol{x}}_{k+1})$ **then**
      Break from loop
    else
      $\hat{L}_{k+1} := r_u \hat{L}_{k+1}$
    end if
  end loop
  $L_{k+1} := \hat{L}_{k+1},\ \boldsymbol{x}_{k+1} := \hat{\boldsymbol{x}}_{k+1}$
  $A_{k+1} := \hat{A}_{k+1},\ \gamma_{k+1} := \hat{\gamma}_{k+1}$
  $\boldsymbol{v}_{k+1} := \frac{1}{\hat{\gamma}_{k+1}}(\gamma_k\boldsymbol{v}_k + \hat{a}_{k+1}(\hat{L}_{k+1} + \mu_\Psi)\hat{\boldsymbol{x}}_{k+1} - \hat{a}_{k+1}(\hat{L}_{k+1} - \mu_f)\hat{\boldsymbol{y}}_{k+1})$
end for
return x_K

algorithm parameters are marked with $\hat{(.)}$ and the updates in which they appear use the $:=$ operator.

E. Convergence Analysis

The convergence of ACGM is governed by (3), with the guarantee given by A_k. The growth rate of A_k is affected by the outcome of the line-search procedure. We formulate a simple lower bound for A_k that accounts for the worst case search behavior. To simplify notation, we introduce the local inverse condition number

$$q_{k+1} \triangleq \frac{\mu}{L'_{k+1}} = \frac{\mu}{L_{k+1} + \mu_\Psi}, \quad k \ge 0.$$

If $L_{k+1} \ge L_f$, then the descent condition for f in (14) holds regardless of the algorithmic state. Therefore, the backtracking search will guarantee that

$$L_{k+1} \le L_u \triangleq \max\{r_u L_f, r_d L_0\}, \quad k \ge 0. \tag{33}$$

Let the worst case local inverse condition number be defined as

$$q_u \triangleq \frac{\mu}{L_u + \mu_\Psi} \le q_{k+1}, \quad k \ge 0.$$

Theorem 2: The convergence guarantee A_k for ACGM is lower bounded in the non-strongly convex case *($\mu = 0$)* by

$$A_k \ge \frac{(k+1)^2}{4L_u}, \quad k \ge 1, \tag{34}$$

and in the strongly convex case *($\mu > 0$)* by

$$A_k \ge \frac{1}{L_u - \mu_f}(1 - \sqrt{q_u})^{-(k-1)}, \quad k \ge 1. \tag{35}$$

Proof: See Appendix E. ■

F. ACGM in Extrapolated Form

An interesting property of FGM is that for all $k \ge 0$, the point $\boldsymbol{y}_{k+2}$ where the gradient is queried during iteration $k+1$ can be expressed in terms of the previous two iterates $\boldsymbol{x}_{k+1}$ and $\boldsymbol{x}_k$ by extrapolation, namely

$$\boldsymbol{y}_{k+2} = \boldsymbol{x}_{k+1} + \beta_{k+1}(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k), \quad k \ge 0,$$

where β_{k+1} is an auxiliary point extrapolation factor. To bring ACGM to a form in which it can be easily compared with FGM, as well as with FISTA and FISTA-CP, we demonstrate that ACGM (Algorithm 2) also exhibits an auxiliary point extrapolation property. The difference is that β_{k+1} can only be computed during iteration $k+1$ due to uncertainties in the outcome of line-search. First, we show the following property of ACGM, which carries over from FGM.

Lemma 2: The estimate function vertices can be obtained from successive iterates through extrapolation as

$$\boldsymbol{v}_{k+1} = \boldsymbol{x}_k + \frac{A_{k+1}}{a_{k+1}}(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k), \quad k \ge 0.$$

Proof: See Appendix F. ■

By combining Lemma 2 with (29) and rearranging terms, we obtain the auxiliary point extrapolation expression for ACGM as

$$\boldsymbol{y}_{k+1} = \boldsymbol{x}_k + \beta_k(\boldsymbol{x}_k - \boldsymbol{x}_{k-1}), \tag{36}$$

where the extrapolation factor β_k is given by

$$\beta_k = \frac{a_{k+1}\gamma_k\left(\frac{A_k}{a_k} - 1\right)}{A_k\gamma_{k+1} + a_{k+1}\gamma_k}, \quad k \ge 1. \tag{37}$$

We denote the *vertex extrapolation factor* in Lemma 2 as

$$t_k \triangleq \begin{cases} \frac{A_k}{a_k}, & k \ge 1, \\ 0, & k = 0. \end{cases} \tag{38}$$

The accumulated weights and the curvature ratios γ_k/γ_{k+1} can be written in terms of t_k for all $k \ge 0$ as

$$A_{k+1} \overset{(30)}{=} \frac{A_{k+1}^2\gamma_{k+1}}{(L_{k+1} + \mu_\Psi)a_{k+1}^2} \overset{(38)}{=} \frac{\gamma_{k+1}t_{k+1}^2}{L_{k+1} + \mu_\Psi}, \tag{39}$$

$$A_0 = 0 \overset{(38)}{=} \frac{\gamma_0 t_0^2}{L_0 + \mu_\Psi}, \tag{40}$$

$$\frac{\gamma_k}{\gamma_{k+1}} \overset{(30)}{=} 1 - \frac{A_{k+1}a_{k+1}\mu}{(L_{k+1} + \mu_\Psi)a_{k+1}^2} \overset{(38)}{=} 1 - q_{k+1}t_{k+1}. \tag{41}$$

Expressions (38), (39), (40), and (41) facilitate the derivation of a recursion rule for t_k that does not depend on either a_k or A_k for all $k \ge 0$ and $\mu \ge 0$ as follows:

$$(L_{k+1} + \mu_\Psi)A_{k+1} - (L_{k+1} + \mu_\Psi)a_{k+1} - \frac{L_{k+1} + \mu_\Psi}{L_k + \mu_\Psi}(L_k + \mu_\Psi)A_k = 0$$

$$\Leftrightarrow \gamma_{k+1}t_{k+1}^2 - \gamma_{k+1}t_{k+1} - \frac{L_{k+1} + \mu_\Psi}{L_k + \mu_\Psi}\gamma_k t_k^2 = 0$$

$$\Leftrightarrow t_{k+1}^2 + t_{k+1}(q_k t_k^2 - 1) - \frac{L_{k+1} + \mu_\Psi}{L_k + \mu_\Psi}t_k^2 = 0. \tag{42}$$

Algorithm 3: ACGM in Extrapolated Form
ACGM$(\boldsymbol{x}_0, L_0, \mu_f, \mu_\Psi, K)$.

$\boldsymbol{x}_{-1} = \boldsymbol{x}_0,\ \mu = \mu_f + \mu_\Psi,\ t_0 = 0,\ q_0 = \frac{\mu}{L_0+\mu_\Psi}$
for $k = 0, \ldots, K-1$ **do**
  $\hat{L}_{k+1} := r_d L_k$
  loop
    $\hat{q}_{k+1} := \frac{\mu}{\hat{L}_{k+1}+\mu_\Psi}$
    $\hat{t}_{k+1} := \frac{1}{2}\left(1 - q_k t_k^2 + \sqrt{(1-q_k t_k^2)^2 + 4\frac{\hat{L}_{k+1}+\mu_\Psi}{L_k+\mu_\Psi} t_k^2}\right)$
    $\hat{\boldsymbol{y}}_{k+1} := \boldsymbol{x}_k + \frac{t_k - 1}{\hat{t}_{k+1}} \frac{1-\hat{q}_{k+1}\hat{t}_{k+1}}{1-\hat{q}_{k+1}} (\boldsymbol{x}_k - \boldsymbol{x}_{k-1})$
    $\hat{\boldsymbol{x}}_{k+1} := \operatorname{prox}_{\frac{1}{\hat{L}_{k+1}}\Psi}\left(\hat{\boldsymbol{y}}_{k+1} - \frac{1}{\hat{L}_{k+1}} \nabla f(\hat{\boldsymbol{y}}_{k+1})\right)$
    if $f(\hat{\boldsymbol{x}}_{k+1}) \leq Q_{f,\hat{L}_{k+1},\hat{\boldsymbol{y}}_{k+1}}(\hat{\boldsymbol{x}}_{k+1})$ **then**
      Break from loop
    else
      $\hat{L}_{k+1} := r_u \hat{L}_{k+1}$
    end if
  end loop
  $\boldsymbol{x}_{k+1} := \hat{\boldsymbol{x}}_{k+1},\ L_{k+1} := \hat{L}_{k+1}$
  $q_{k+1} := \hat{q}_{k+1},\ t_{k+1} := \hat{t}_{k+1}$
end for
return x_K

Lastly, we write down the auxiliary point extrapolation factor β_k in (37) as

$$\beta_k \overset{(38)}{=} \frac{t_k - 1}{t_{k+1}} \frac{A_{k+1}\gamma_k}{A_k\gamma_{k+1} + a_{k+1}\gamma_k} \overset{(26)}{=} \frac{t_k - 1}{t_{k+1}} \frac{\frac{\gamma_k}{\gamma_{k+1}}}{1 - \frac{\mu a_{k+1}^2}{A_{k+1}\gamma_{k+1}}}$$
$$\overset{(41)}{=} \frac{t_k - 1}{t_{k+1}} \frac{1 - q_{k+1}t_{k+1}}{1 - q_{k+1}}, \quad k \geq 1. \tag{43}$$

Since $\boldsymbol{x}_0 = \boldsymbol{v}_0$, from (29) we always have that $\boldsymbol{y}_1 = \boldsymbol{x}_0$. Therefore, to be able to use (36) during the first iteration $k = 0$, we have to define $\boldsymbol{x}_{-1} \triangleq \boldsymbol{x}_0$. Parameter β_0 can take any real value in (36). For simplicity, we choose to compute β_0 using (43) with $k = 0$.

Now, from (42) and (43), we can formulate ACGM based on extrapolation, as presented in Algorithm 3. Note that Algorithms 2 and 3 differ only in form. They are theoretically guaranteed to produce identical iterates.

IV. Wall-Clock Time Units

When measuring the convergence rate, the prevailing indexing strategies for objective values found in the literature are based on either iterations (e.g., [8], [15]–[17]), running time in a particular computing environment (e.g., [8], [17]), or the number of calls to a low-level routine that dominates all others in complexity (e.g., [15], [23]). The first approach cannot cope with the diversity of methods studied. For example, AMGS makes two gradient steps per iteration whereas FISTA makes only one. The latter two approaches do not generalize to the entire problem class. Running time, in particular, is highly sensitive to system architecture and implementation details. For instance, inadequate cache utilization can increase running time by at least an order of magnitude [24].

Optimization algorithms must also take into account the constraints determined by computer hardware technology, especially the limitation on microprocessor frequency imposed by power consumption and generated heat [24]. This restriction, along with the increase in magnitude of large-scale problems, has rendered serial machines unsuitable for the computation of large-scale oracle functions. Therefore, large-scale optimization algorithms need to be executed on parallel systems. To account for parallelism, we extend the oracle model by introducing the following abstraction. We assume that each oracle function call is processed by a dedicated parallel processing unit (PPU). A PPU may be itself a collection processors. While we do not set a limit on the number of processors a single PPU may have,[3] we do assume that all PPUs are identical. For instance, a PPU may be a single central processing unit (CPU) core or a collection of graphics processing unit (GPU) cores. Since the exact implementation of the oracle functions need not be known to the optimization algorithm, the manner in which processors *within* a PPU are utilized need not be known as well. However, on a higher level of abstraction, we are able to explicitly execute an unlimited number of oracle functions simultaneously, as long as there are no race conditions. Throughout this work, we consider only this shared memory parallel model.

To account for the broadness of the problem class, wherein oracle functions may or may not be separable[4] and their relative cost may vary, we impose that the complexity of computing $f(\boldsymbol{x})$ is comparable to that of $\nabla f(\boldsymbol{x})$ [25]. We denote the amount of wall-clock time required to evaluate $f(\boldsymbol{x})$ or $\nabla f(\boldsymbol{x})$ by 1 wall-clock time unit (WTU). In many applications, the two calls share subexpressions. However, for a given value of $\boldsymbol{x}$, $f(\boldsymbol{x})$ and $\nabla f(\boldsymbol{x})$ are computed simultaneously on separate PPUs, which merely reduces the cost of a WTU without violating the oracle model. Because we are dealing with large-scale problems and Ψ is assumed to be simple, we attribute a cost of 0 WTU to $\Psi(\boldsymbol{x})$ and $\operatorname{prox}_{\tau\Psi}(\boldsymbol{x})$ calls as well as to individual scalar-vector multiplications and vector additions [4].

In the following, we analyze the resource usage and runtime behavior of FGM, AMGS, FISTA, FISTA-CP, and ACGM under the above assumptions. FGM and FISTA-CP compute at every iteration $k \geq 0$ the gradient at the auxiliary point $(\nabla f(\boldsymbol{y}_{k+1}))$ but lack an explicit line-search scheme. The per-iteration cost of these methods is therefore always 1 WTU. For methods that employ line-search, parallelization involves the technique of speculative execution [24] whereby the validation phase of the search takes place in parallel with the advancement phase of the next iteration. When a backtrack occurs, function and gradient values of points that change have to be recomputed, stalling the entire multi-threaded system accordingly. It follows that additional backtracks have the same cost. If the search parameters are tuned properly, most iterations do not have backtracks.

AMGS requires at iteration k calls to both $\nabla f(\boldsymbol{y}_{k+1})$ and $\nabla f(\boldsymbol{x}_{k+1})$. Iterate $\boldsymbol{x}_{k+1}$ can only be computed after $\nabla f(\boldsymbol{y}_{k+1})$ completes and the next auxiliary point $\boldsymbol{y}_{k+2}$ requires $\nabla f(\boldsymbol{x}_{k+1})$. Hence, an iteration without backtracks entails 2 WTU. A backtrack at iteration k involves the recalculation of $\nabla f(\boldsymbol{y}_{k+1})$, which means that each backtrack also costs 2 WTU.

FISTA advances using one $\nabla f(\boldsymbol{y}_{k+1})$ call. The values of $f(\boldsymbol{y}_{k+1})$ and $f(\boldsymbol{x}_{k+1})$ are only needed to validate the Lipschitz estimate. The $f(\boldsymbol{y}_{k+1})$ call can be performed in parallel

[3] In practice, the limit on the number of execution threads is imposed by the communication and synchronization overhead, which varies widely between implementations.

[4] For instance, a single matrix-vector multiplication is separable (with respect to individual scalar operations) whereas a chain of such multiplications is not.

TABLE II
PER-ITERATION COST IN WTU OF LINE-SEARCH METHODS AMGS, FISTA, AND ACGM

Iteration phase	AMGS	FISTA	ACGM
Iteration without backtrack	2	1	1
Each backtrack	2	1	2

with $\nabla f(\boldsymbol{y}_{k+1})$ but the calculation of $\boldsymbol{x}_{k+1}$ utilizes $\nabla f(\boldsymbol{y}_{k+1})$. The backtracking strategy of FISTA does not require the recalculation of $\boldsymbol{y}_{k+1}$ and its oracle values. However, the need for a backtrack can only be asserted after the completion of $f(\boldsymbol{x}_{k+1})$. Therefore, an iteration without backtracks of FISTA entails 1 WTU, with each backtrack adding 1 WTU to the cost.

The ability of ACGM to decrease the Lipschitz estimate necessitates the recalculation of $\boldsymbol{y}_{k+1}$, in addition to the delay in the backtrack condition assessment. As a result, ACGM has an iteration base cost of 1 WTU and a 2 WTU backtrack cost. Note that the Algorithm 2 and Algorithm 3 forms of ACGM are identical with respect to WTU usage. The iteration costs of AMGS, FISTA, and ACGM are summarized in Table II.

Interestingly, the above algorithms need at most three concurrent high-level computation threads (PPUs) to operate. The assignment of different computations to different PPUs at every time unit, along with the iteration that computation are detailed in Table III for an iteration $k \geq 1$ without backtracks and in Table IV for an iteration where a single backtrack occurs. The behavior of subsequent backtracks follows closely the pattern shown in Table IV.

V. SIMULATION RESULTS

We test ACGM against the state-of-the-art methods on a typical non-strongly convex inverse problem in Subsection V-A whereas in Subsection V-B we focus on a strongly convex machine learning problem. Both applications feature l_1-norm regularization [26]. They have been chosen due to their popularity and simplicity. While effective approaches that exploit additional problem structure, such as sparsity of optimal points, have been proposed in the literature (e.g. [10]–[13]), we consider the applications studied in this section as representative of a broader class of problems to which the above specialized methodologies may not apply.

A. l_1-regularized Image Deblurring

To better compare the capabilities of ACGM (Algorithm 3) to those of FISTA, we choose the very problem FISTA was introduced to solve, namely the l_1-regularized deblurring of images.[5] For ease and accuracy of benchmarking, we have adopted the experimental setup from Section 5.1 in [7]. Here, the composite objective function is given by

$$f(\boldsymbol{x}) = \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2, \qquad \Psi(\boldsymbol{x}) = \lambda\|\boldsymbol{x}\|_1,$$

where $\boldsymbol{A} = \boldsymbol{R}\boldsymbol{W}$. The linear operator $\boldsymbol{R}$ is a Gaussian blur with standard deviation 4.0 and a 9×9 pixel kernel, applied using reflexive boundary conditions [28]. The linear operator $\boldsymbol{W}$ is the inverse three-stage Haar wavelet transform. The digital image $\boldsymbol{x} \in \mathbb{R}^{n_1 \times n_2}$ has dimensions $n_1 = n_2 = 256$. The blurred image $\boldsymbol{b}$ is obtained by applying $\boldsymbol{R}$ to the cameraman test image [7] with pixel values scaled to the $[0, 1]$ range, followed by the addition of Gaussian noise (zero-mean, standard deviation 10^{-3}). The constant L_f can be computed as the maximum eigenvalue of a symmetric Toeplitz-plus-Hankel matrix (more details in [28]), which yields a value of $L_f = 2.0$. The problem is non-strongly convex with $\mu = \mu_f = \mu_\Psi = 0$. The regularization parameter λ is set to $2 \cdot 10^{-5}$ to account for the noise level of $\boldsymbol{b}$.

We have noticed that several monographs in the field (e.g. [8], [16]) do not include AMGS in their benchmarks. For completeness, we compare Algorithm 3 against both FISTA with backtracking line-search (FISTA-BT) and AMGS. The starting point $\boldsymbol{x}_0$ was set to $\boldsymbol{W}^{-1}\boldsymbol{b}$ for all algorithms. AMGS and FISTA were run using $r_u^{\mathrm{AMGS}} = r_u^{\mathrm{FISTA}} = 2.0$ and $r_d^{\mathrm{AMGS}} = 0.9$ as these values were suggested in [23] to "provide good performance in many applications". Assuming that most of time the Lipschitz constant estimates hover around a fixed value, we have for AMGS that a backtrack occurs every $-(\log r_u^{\mathrm{AMGS}})/(\log r_d^{\mathrm{AMGS}})$ iterations. The cost ratio between a backtrack and an iteration without backtracks for ACGM is double that of AMGS. Therefore, to ensure that the line-search procedures of both methods have comparable computational overheads, we have chosen $r_u^{\mathrm{ACGM}} = r_u^{\mathrm{AMGS}}$ and $r_d^{\mathrm{ACGM}} = \sqrt{r_d^{\mathrm{AMGS}}}$.

To showcase the importance of employing an algorithm with an efficient and robust line-search procedure, we have considered two scenarios: a normally underestimated initial guess $L_0 = 0.3L_f$ (Fig. 1) and a greatly overestimated $L_0 = 10L_f$. The convergence rate is measured as the difference between the objective function values and an optimal value *estimate* $F(\hat{\boldsymbol{x}}^*)$, where $\hat{\boldsymbol{x}}^*$ is the iterate obtained after running fixed step size FISTA with the correct Lipschitz constant parameter for 10000 iterations.

When indexing in iterations (Figs. 1(a) and 1(d)), ACGM converges roughly as fast as AMGS. ACGM takes the lead after 500 iterations, owing mostly to the superiority of ACGM's descent condition over AMGS's stringent "damped relaxation condition" [15]. When indexed in WTU, ACGM clearly surpasses AMGS from the very beginning (Figs. 1(b) and 1(e)), because of ACGM's low per-iteration complexity.

FISTA-BT lags behind in the overestimated case, regardless of the convergence measure (Figs. 1(d) and 1(e)), and it is also slightly slower in the underestimated case (Figs. 1(a) and 1(b)). The disadvantage of FISTA-BT lies in the inability of its line-search procedure to decrease the Lipschitz constant estimate while the algorithm is running. Consequently, in both cases, FISTA-BT produces on average a higher Lipschitz estimate than ACGM. This is clearly evidenced by Figs. 1(c) and 1(f).

B. Logistic Regression With Elastic Net

As a strongly convex application, we choose a randomly generated instance of the logistic regression classification task [29], regularized with an elastic net [30]. The objective function components are given by

$$f(\boldsymbol{x}) = -\langle \boldsymbol{y}, \boldsymbol{A}\boldsymbol{x}\rangle + \sum_{i=1}^{m} \log\left(1 + e^{\langle \boldsymbol{a}_i^T, \boldsymbol{x}\rangle}\right),$$

$$\Psi(\boldsymbol{x}) = \lambda_1\|\boldsymbol{x}\|_1 + \frac{\lambda_2}{2}\|\boldsymbol{x}\|_2^2,$$

where the matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ has rows $\boldsymbol{a}_i^T$, $i \in \{1, \ldots, m\}$, $\boldsymbol{y} \in \mathbb{R}^m$ is the vector of classification labels and the elastic net

[5] A particular case of ACGM in estimate sequence form, designed only for non-strongly convex objectives, was tested on the same problem in [27].

TABLE III
RESOURCE ALLOCATION AND RUNTIME BEHAVIOR OF PARALLEL BLACK-BOX FGM, FISTA-CP, AMGS, FISTA, AND ACGM WHEN NO BACKTRACKS OCCUR (ITERATION $k \geq 1$ STARTS AT TIME T)

Method	WTU	PPU 1		PPU 2		PPU 3	
		Comp.	Iter.	Comp.	Iter.	Comp.	Iter.
FGM	T	$\nabla f(\boldsymbol{y}_{k+1})$	k	Idle		Idle	
	T + 1	$\nabla f(\boldsymbol{y}_{k+2})$	k + 1	Idle		Idle	
FISTA-CP	T	$\nabla f(\boldsymbol{y}_{k+1})$	k	Idle		Idle	
	T + 1	$\nabla f(\boldsymbol{y}_{k+2})$	k + 1	Idle		Idle	
AMGS	T	$\nabla f(\boldsymbol{y}_{k+1})$	k	Idle		Idle	
	T + 1	$\nabla f(\boldsymbol{x}_{k+1})$	k	Idle		Idle	
	T + 2	$\nabla f(\boldsymbol{y}_{k+2})$	k + 1	Idle		Idle	
FISTA	T	$\nabla f(\boldsymbol{y}_{k+1})$	k	$f(\boldsymbol{y}_{k+1})$	k	$f(\boldsymbol{x}_k)$	k - 1
	T + 1	$\nabla f(\boldsymbol{y}_{k+2})$	k + 1	$f(\boldsymbol{y}_{k+2})$	k + 1	$f(\boldsymbol{x}_{k+1})$	k
	T + 2	$\nabla f(\boldsymbol{y}_{k+3})$	k + 2	$f(\boldsymbol{y}_{k+3})$	k + 2	$f(\boldsymbol{x}_{k+2})$	k + 1
ACGM	T	$\nabla f(\boldsymbol{y}_{k+1})$	k	$f(\boldsymbol{y}_{k+1})$	k	$f(\boldsymbol{x}_k)$	k - 1
	T + 1	$\nabla f(\boldsymbol{y}_{k+2})$	k + 1	$f(\boldsymbol{y}_{k+2})$	k + 1	$f(\boldsymbol{x}_{k+1})$	k
	T + 2	$\nabla f(\boldsymbol{y}_{k+3})$	k + 2	$f(\boldsymbol{y}_{k+3})$	k + 2	$f(\boldsymbol{x}_{k+2})$	k + 1

TABLE IV
RESOURCE ALLOCATION AND RUNTIME BEHAVIOR OF PARALLEL BLACK-BOX AMGS, FISTA, AND ACGM WHEN A SINGLE BACKTRACK OCCURS (ITERATION $k \geq 1$ STARTS AT TIME T)

Method	WTU	PPU 1		PPU 2		PPU 3	
		Comp.	Iter.	Comp.	Iter.	Comp.	Iter.
AMGS	T	$\nabla f(\boldsymbol{y}_{k+1})$	k	Idle		Idle	
	T + 1	$\nabla f(\boldsymbol{x}_{k+1})$	k	Idle		Idle	
	T + 2	$\nabla f(\boldsymbol{y}_{k+1})$	k	Idle		Idle	
	T + 3	$\nabla f(\boldsymbol{x}_{k+1})$	k	Idle		Idle	
	T + 4	$\nabla f(\boldsymbol{y}_{k+2})$	k + 1	Idle		Idle	
FISTA	T	$\nabla f(\boldsymbol{y}_{k+1})$	k	$f(\boldsymbol{y}_{k+1})$	k	$f(\boldsymbol{x}_k)$	k - 1
	T + 1	$\nabla f(\boldsymbol{y}_{k+2})$	k + 1	$f(\boldsymbol{y}_{k+2})$	k + 1	$f(\boldsymbol{x}_{k+1})$	k
	T + 2	$\nabla f(\boldsymbol{y}_{k+2})$	k + 1	$f(\boldsymbol{y}_{k+2})$	k + 1	$f(\boldsymbol{x}_{k+1})$	k
	T + 3	$\nabla f(\boldsymbol{y}_{k+3})$	k + 2	$f(\boldsymbol{y}_{k+3})$	k + 2	$f(\boldsymbol{x}_{k+2})$	k + 1
ACGM	T	$\nabla f(\boldsymbol{y}_{k+1})$	k	$f(\boldsymbol{y}_{k+1})$	k	$f(\boldsymbol{x}_k)$	k - 1
	T + 1	$\nabla f(\boldsymbol{y}_{k+2})$	k + 1	$f(\boldsymbol{y}_{k+2})$	k + 1	$f(\boldsymbol{x}_{k+1})$	k
	T + 2	$\nabla f(\boldsymbol{y}_{k+1})$	k	$f(\boldsymbol{y}_{k+1})$	k	Idle	
	T + 3	$\nabla f(\boldsymbol{y}_{k+2})$	k + 1	$f(\boldsymbol{y}_{k+2})$	k + 1	$f(\boldsymbol{x}_{k+1})$	k
	T + 4	Idle		Idle		$f(\boldsymbol{x}_{k+2})$	k + 1

regularizer Ψ has parameters λ_1 and λ_2. The problem size is $m = n = 10000$. The matrix A is sparse and has 10% of its elements non-zero, each sampled as independent and identically distributed (i.i.d.) from the standard Gaussian distribution $\mathcal{N}(0, 1)$. The labels $\boldsymbol{y}_i$ are randomly generated with probability

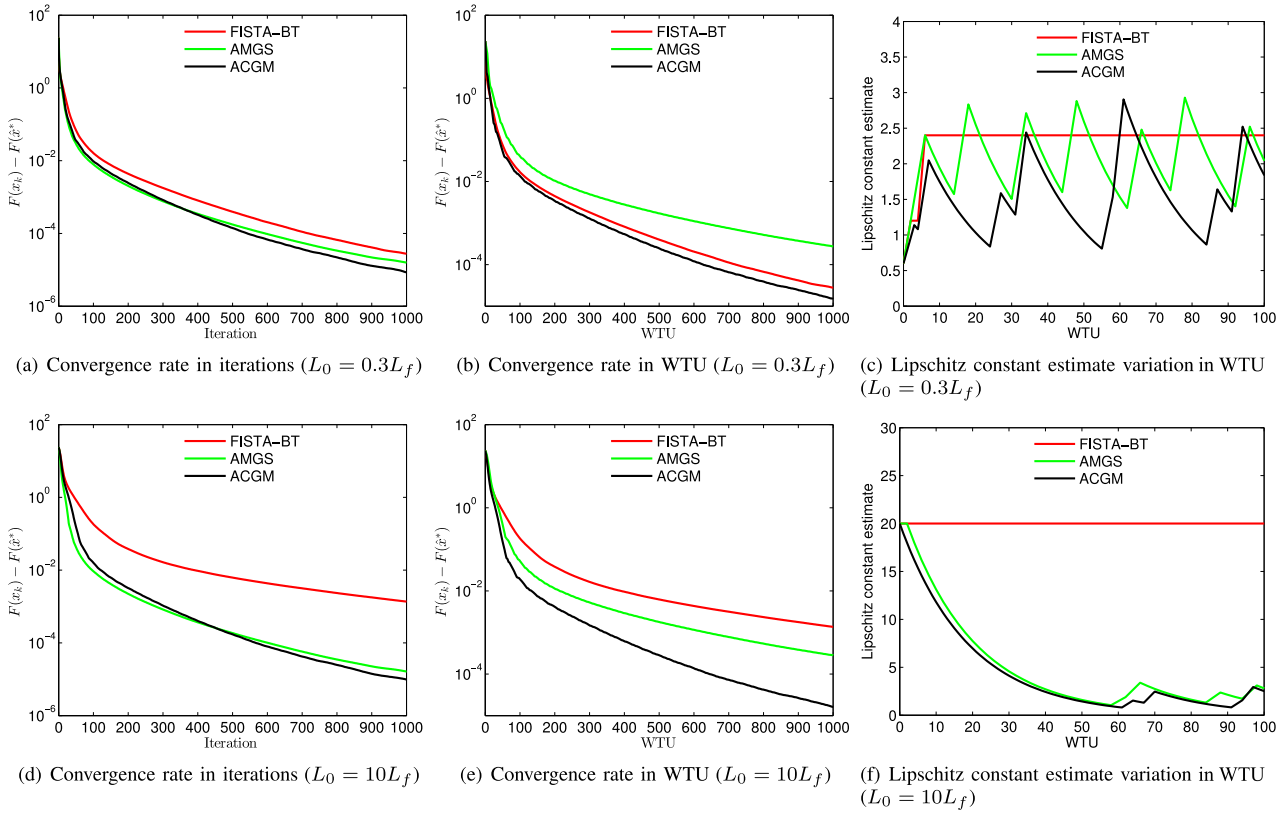
$$\mathbb{P}(\boldsymbol{Y}_i = 1) = \frac{1}{1 + e^{\langle \boldsymbol{a}_i^T, \boldsymbol{x} \rangle}}, \quad i \in \{1, \ldots, m\}.$$

The gradient of function f has a global Lipschitz constant $L_\sigma = \frac{1}{4}\sigma_{max}(\boldsymbol{A})^2$, where $\sigma_{max}(\boldsymbol{A})$ is the largest singular value of $\boldsymbol{A}$. The computation of $\sigma_{max}(\boldsymbol{A})$ is generally intractable for large-scale problems and optimization algorithms need instead to rely on an estimate of this value. The smooth part f is *not* strongly convex ($\mu_f = 0$). The elastic net parameters are $\lambda_1 = 1$ and $\lambda_2 = 10^{-3} L_\sigma$. Hence $\mu = \mu_\Psi = \lambda_2$. Elastic net regularization is specified by the user [30] and we assume that optimization algorithms can access μ_Ψ.

We benchmark ACGM against methods that have convergence guarantees. These methods are either equipped with a line-search procedure, such as FISTA and AMGS, or rely on L_f being known in advance, namely FISTA-CP and MOS. We do not include scAPG in our benchmark because $\mu_f = 0$. We also do not consider methods that owe their performance on specific applications to heuristic improvements that either significantly degrade the provable convergence rate, such as in AA (see Appendix A for proof), or invalidate it altogether, like adaptive restart in FISTA [31] or in AA [18].

The starting point $\boldsymbol{x}_0$, the same for all algorithms tested, has entries randomly sampled as i.i.d. from $\mathcal{N}(0, 1)$. For the same reasons as outlined in Subsection V-A, we have chosen $r_u^{\mathrm{ACGM}} = r_u^{\mathrm{AMGS}} = r_u^{\mathrm{FISTA}} = 2.0$, $r_d^{\mathrm{AMGS}} = 0.9$, and $r_d^{\mathrm{ACGM}} = \sqrt{r_d^{\mathrm{AMGS}}}$.

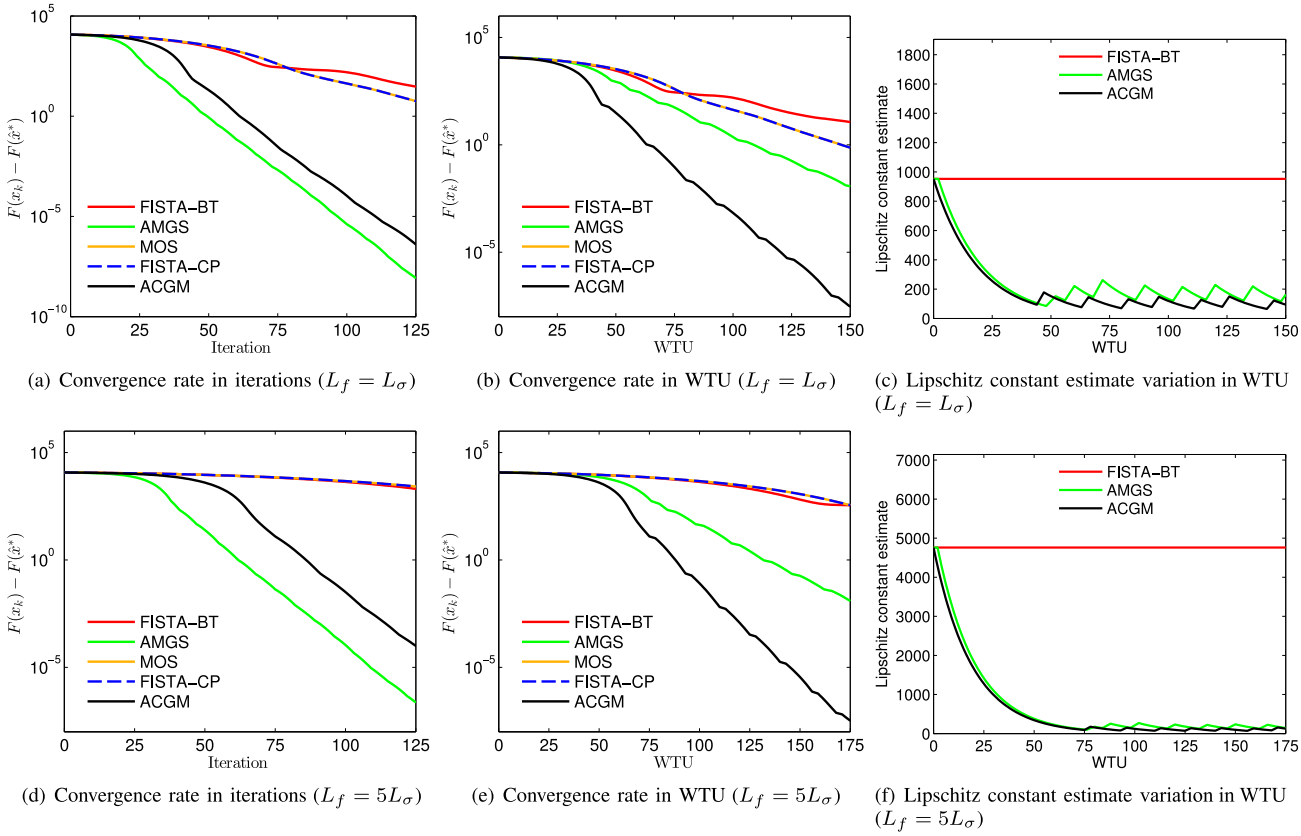
We have computed the optimal point estimate $\hat{\boldsymbol{x}}^*$ as the iterate with the smallest objective value obtained after running AMGS for 500 iterations using $L_f = L_\sigma$ with the other parameters as mentioned above. Methods equipped with a line-search procedure incur a search overhead whereas the other methods do not. For fair comparison, we have tested the collection of



(a) Convergence rate in iterations ($L_0 = 0.3L_f$) (b) Convergence rate in WTU ($L_0 = 0.3L_f$) (c) Lipschitz constant estimate variation in WTU ($L_0 = 0.3L_f$)

(d) Convergence rate in iterations ($L_0 = 10L_f$) (e) Convergence rate in WTU ($L_0 = 10L_f$) (f) Lipschitz constant estimate variation in WTU ($L_0 = 10L_f$)

Fig. 1. Convergence results on the l_1-regularized image deblurring problem ($\mu = 0$).



(a) Convergence rate in iterations ($L_f = L_\sigma$) (b) Convergence rate in WTU ($L_f = L_\sigma$) (c) Lipschitz constant estimate variation in WTU ($L_f = L_\sigma$)

(d) Convergence rate in iterations ($L_f = 5L_\sigma$) (e) Convergence rate in WTU ($L_f = 5L_\sigma$) (f) Lipschitz constant estimate variation in WTU ($L_f = 5L_\sigma$)

Fig. 2. Convergence results on logistic regression with elastic net ($\mu = 10^{-3}L_\sigma$).

TABLE V
FEATURES OF BLACK-BOX FIRST-ORDER METHODS

Feature	Prox. point	FGM	AMGS	FISTA	FISTACP	MOS	scAPG	ACGM
Composite objective	yes	no	yes	yes	yes	yes	partial	yes
Line-search	no	no	yes	partial	no	no	yes	yes
$\mathcal{O}(\frac{1}{k^2})$ rate for $\mu = 0$	no	yes	yes	yes	yes	yes	no	yes
Linear rate for $\mu > 0$	yes	yes	yes	no	yes	yes	yes	yes
$\mathcal{O}((1-\sqrt{q})^k)$ rate for $\mu > 0$	no	yes	no	no	yes	no	yes	yes

methods in the accurate $L_f = L_\sigma$ case as well as the overestimated $L_f = 5L_\sigma$ case (Fig. 2).

When indexing in iterations, AMGS converges the fastest (Figs. 2(a) and 2(d)). However, AMGS has the same asymptotic rate (in iterations) as ACGM, despite AMGS performing around twice the number of proximal gradient steps per iteration. While proximal gradient steps (incurring 1 WTU each) in AMGS improve the Lipschitz constant estimate (Fig. 2(c)), they do not appear to be used efficiently in advancing the algorithm. Therefore, AMGS is inferior to ACGM and FISTA-CP in terms of WTU usage (Figs. 2(b) and 2(e)). Note that FISTA-CP and MOS display nearly identical convergence behaviors (Figs. 2(a), 2(b), 2(d), and 2(e)), as theoretically argued in Appendix A.

This particular application emphasizes the importance of taking into account the local curvature of the function. Whereas ACGM and FISTA-CP have identical a priori worst-case rates, FISTA-CP (and consequently MOS) lags behind considerably, even when an accurate value of L_f is supplied (Figs. 2(a) and 2(b)). The reason is that the Lipschitz estimates of ACGM are several times smaller than the global value L_f (Fig. 2(c)). The difference between local and global curvature is so great that FISTA-CP's ability to exploit strong convexity does not give it a sizable performance advantage over FISTA on this problem.[6] The benefit of ACGM's line-search is predictably more evident in the inaccurate case (Fig. 2(f)). The estimates produced by the AMGS's damped relaxation condition are considerably higher than those of ACGM, further contributing to ACGM's superior convergence behavior in WTU (Figs. 2(b) and 2(e)).

VI. DISCUSSION AND CONCLUSIONS

The proposed method, ACGM, when formulated using extrapolation, encompasses several existing optimization schemes. Specifically, Algorithm 3 without the line-search procedure, i.e., with $L_k = L_f$ for all $k \geq 0$, produces the same iterates as FISTA-CP with the theoretically optimal step size $\tau^{\mathrm{FISTA-CP}} = \frac{1}{L_f}$. In the non-strongly convex case, ACGM without line-search reduces to constant step size FISTA. Also for $\mu = 0$, ACGM with line-search constitutes a simplified and more intuitive alternative to a recently introduced (without derivation) line-search extension of FISTA [32].

However, ACGM is more than an umbrella method. ACGM's generality and unique collection of features is a strength in itself. For instance, FISTA suffers from two drawbacks: the parameter t_k^{FISTA} update is oblivious to the change in local curvature and the Lipschitz constant estimates cannot decrease. Hence, if the initial Lipschitz estimate is erroneously large, FISTA will slow down considerably (exemplified in Subsection V-A). We formally express the advantages of ACGM's line-search over that of FISTA in the following proposition.

Proposition 2: In the non-strongly convex case ($\mu = 0$), under identical local curvature conditions, when $r_u^{\mathrm{ACGM}} = r_u^{\mathrm{FISTA}}$, ACGM has superior theoretical convergence guarantees to FISTA, namely

$$A_k^{\mathrm{ACGM}} \geq A_k^{\mathrm{FISTA}}, \quad k \geq 0.$$

Proof: See Appendix G. ■

The ability to dynamically and frequently adjust to the local Lipschitz constant gives ACGM an advantage over FISTA-CP as well, even when an accurate estimate of the Lipschitz constant is available beforehand (illustrated in Subsection V-B). The advantage over MOS is even greater since MOS is slightly slower than FISTA-CP (see Appendix A). The scAPG method is similar to ACGM, but only when $\mu_f > 0$ and $\boldsymbol{x}_0$ is feasible. We leave the generalization of ACGM to encompass scAPG, and thus expand its range of applications, as a topic for future research.

ACGM is also theoretically guaranteed to outperform AMGS, as argued in Appendix A. The per-iteration complexity of ACGM, both in the non-strongly and strongly convex cases ($\mu \geq 0$), lies well below that of AMGS. Considering that backtracks rarely occur, it approaches that of FISTA (see Table II) and the absolute minimum of 1 WTU per iteration.

Thus, this is the first time, as far as we are aware, that a method has been shown to be superior, from theoretical as well as simulation results (Section V), to AMGS, FISTA, and FISTA-CP. The aforementioned features of ACGM are summarized and compared to those of the competing black-box first-order methods in Table V. As can be discerned from Table V, ACGM is the only method of its class that is able to *combine* the strengths of AMGS (generality) and FGM (speed). The superiority of ACGM stems from this unique combination.

Furthermore, due to its robustness, ACGM is not only applicable to the entire composite problem class, where the Lipschitz constant may not be known, but is also able to converge on problems where the Lipschitz property of the gradient can be proven to hold only *locally*.

Alongside of a new algorithm, in this work we have provided a means of designing algorithms. We have demonstrated that the estimate sequence concept can be extended to problems outside its original scope. The augmented estimate sequence actually links the concepts of estimate sequence and Lyapunov function, and further argues that both are effective tools not only for the analysis but also for the *design* of fast algorithms. Whether

[6] We forward the reader to [8] for a more detailed comparison between FISTA and FISTA-CP.

augmentation leads to efficient algorithms applicable to other problem classes is a promising topic for future research.

Appendix A
The Asymptotic Convergence Guarantees of Accelerated Black-box First-order Methods

To be able to compare the provable convergence rates of the state-of-the-art black-box methods introduced in Section I, we consider the largest problem class to which they are applicable, namely the class of composite problems with L_f known in advance. For ease of analysis, we study ACGM, AMGS, and scAPG without line-search. This setup does not assume any particular parallel implementation. Therefore, the results in this section are of fundamental theoretical importance.

The asymptotic rate of ACGM matches those of FISTA-CP, scAPG (for strongly convex f and non-strongly convex Ψ), and FGM (when $\Psi = 0$). Hence, we limit our analysis to ACGM, MOS, AMGS, and AA.

In the non-strongly convex case, the convergence guarantees are, respectively, given for all $k \geq 1$ by

$$A_k^{\mathrm{ACGM}} = A_i^{\mathrm{ACGM}} \geq \frac{(k+1)^2}{4L_f} = \frac{(i+1)^2}{4L_f},$$

$$A_k^{\mathrm{MOS}} = A_i^{\mathrm{MOS}} \geq \frac{k^2}{4L_f} = \frac{i^2}{4L_f},$$

$$A_k^{\mathrm{AMGS}} = A_{\frac{i}{2}}^{\mathrm{AMGS}} \geq \frac{k^2}{2L_f} = \frac{i^2}{8L_f},$$

$$A_k^{\mathrm{AA}} = A_{\frac{i}{2}}^{\mathrm{AA}} \geq \frac{k^2}{4L_f} = \frac{i^2}{16L_f},$$

where i gives the number of WTU required by the first k iterations. It trivially follows that

$$\frac{A_i^{\mathrm{ACGM}}}{i^2} \gtrapprox \frac{A_i^{\mathrm{MOS}}}{i^2} > \frac{A_i^{\mathrm{AMGS}}}{i^2} > \frac{A_i^{\mathrm{AA}}}{i^2}, \quad i \geq 2. \tag{44}$$

In the strongly convex case, let q be the inverse condition number of the objective function, $q \triangleq \frac{\mu}{L_f + \mu_\Psi}$. We assume that $q < 1$ since for $q = 1$ the optimization problem can be solved exactly, using only one proximal gradient step. When employing AMGS, Nesterov suggests in [15] either to transfer all strong convexity from f to Ψ, or to restart the algorithm at regular intervals.[7] Both enhancements have the same effect on the convergence guarantee, which can be expressed as

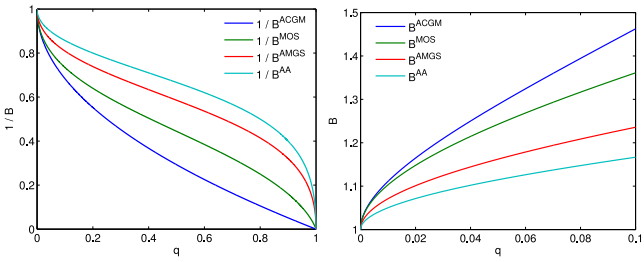
$$A_k^{\mathrm{AMGS}} = A_{\frac{i}{2}}^{\mathrm{AMGS}} \geq C^{\mathrm{AMGS}} \left(B^{\mathrm{AMGS}}\right)^i,$$

where B^{AMGS} is a base signifying the asymptotic convergence rate, given by

$$B^{\mathrm{AMGS}} \triangleq \left(1 + \sqrt{\frac{\mu}{2(L_f - \mu_f)}}\right)^2 = \left(1 + \sqrt{\frac{q}{2(1-q)}}\right)^2,$$

and C^{AMGS} is a proportionality constant.

[7]These suggestions are made in the context of smooth constrained optimization but also apply to composite problems.



Inverse rates as a function of q Rates for $q \leq 0.1$

Fig. 3. Asymptotic rates of ACGM, MOS, AMGS, and AA.

For ACGM, MOS, and AA, we have

$$A_k^{\mathrm{ACGM}} = A_i^{\mathrm{ACGM}} \geq C^{\mathrm{ACGM}} \left(B^{\mathrm{ACGM}}\right)^i,$$

$$A_k^{\mathrm{MOS}} = A_i^{\mathrm{MOS}} \geq C^{\mathrm{MOS}} \left(B^{\mathrm{MOS}}\right)^i,$$

$$A_k^{\mathrm{AA}} = A_{\frac{i}{2}}^{\mathrm{AA}} \geq C^{\mathrm{AA}} \left(B^{\mathrm{AA}}\right)^i,$$

where

$$B^{\mathrm{ACGM}} \triangleq \frac{1}{1 - \sqrt{q}},$$

$$B^{\mathrm{MOS}} \triangleq \left(1 + \frac{1}{2}\sqrt{\frac{q}{1-q}}\right)^2,$$

$$B^{\mathrm{AA}} \triangleq 1 + \frac{1}{2}\sqrt{\frac{q}{1-q}}.$$

Assumption $0 < q < 1$ implies that

$$B^{\mathrm{ACGM}} > B^{\mathrm{MOS}} > B^{\mathrm{AMGS}} > B^{\mathrm{AA}}. \tag{45}$$

A quantitative comparison of the rates can be found in Fig. 3. The inverse rates are compared for every possible value of q in Fig. 3(a) whereas the rates are compared directly in Fig. 3(b) for the range of q found in the vast majority of practical applications.

It can be clearly discerned from (44), (45), and Fig. 3 that ACGM is asymptotically more efficient than MOS, AMGS, and AA, in that order. AMGS is considerably slower than ACGM due to its computationally expensive line-search procedure. By removing line-search, MOS achieves a rate similar to ACGM in the non-strongly convex case and a lower rate (yet comparable when $q \ll 1$) for strongly convex objectives. This, however, comes at the expense of reduced functionality. The heuristic search of AA incurs an extra 1 WTU per iteration without provably advancing the algorithm, explaining why AA has the worst guarantees of the methods studied.

Appendix B
Proof of Proposition 1

By expanding $Q_{f',L+\mu_\Psi,\boldsymbol{y}}(\boldsymbol{x})$ using the definition of Q in (1) and the strong convexity transfer in (12) we obtain

$$Q_{f',L+\mu_\Psi,\boldsymbol{y}}(\boldsymbol{x}) = f(\boldsymbol{y}) + \frac{\mu_\Psi}{2}\|\boldsymbol{y} - \boldsymbol{x}_0\|_2^2$$

$$+ \langle \nabla f(\boldsymbol{y}) + \mu_\Psi(\boldsymbol{y} - \boldsymbol{x}_0), \boldsymbol{x} - \boldsymbol{y}\rangle + \frac{L + \mu_\Psi}{2}\|\boldsymbol{x} - \boldsymbol{y}\|_2^2$$

$$
\begin{aligned}
&- \frac{\mu_\Psi}{2}\|\boldsymbol{x}-\boldsymbol{x}_0\|_2^2 + \frac{\mu_\Psi}{2}\|\boldsymbol{x}-\boldsymbol{x}_0\|_2^2 \\
&= f(\boldsymbol{y}) + \langle \nabla f(\boldsymbol{y}), \boldsymbol{x}-\boldsymbol{y}\rangle + \frac{L}{2}\|\boldsymbol{x}-\boldsymbol{y}\|_2^2 + \frac{\mu_\Psi}{2}\|\boldsymbol{x}-\boldsymbol{x}_0\|_2^2,
\end{aligned} \tag{46}
$$

for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ and $L > 0$. Rewriting (46) based on (1) completes the proof.

Appendix C
Proof of Lemma 1

From the strong convexity property of f', we have a supporting generalized parabola at $\boldsymbol{y}_{k+1}$, given by

$$
f'(\boldsymbol{x}) \geq f'(\boldsymbol{y}_{k+1}) + \langle \nabla f'(\boldsymbol{y}_{k+1}), \boldsymbol{x}-\boldsymbol{y}_{k+1}\rangle + \frac{\mu}{2}\|\boldsymbol{x}-\boldsymbol{y}_{k+1}\|_2^2, \tag{47}
$$

for all $\boldsymbol{x} \in \mathbb{R}^n$ and $k \geq 0$. The first-order optimality condition of (2) implies that there exists a subgradient $\boldsymbol{\xi}$ of function Ψ' at point $\boldsymbol{x}_{k+1}$ such that

$$
g_{L'_{k+1}}(\boldsymbol{y}_{k+1}) = \nabla f'(\boldsymbol{y}_{k+1}) + \boldsymbol{\xi}, \quad k \geq 0.
$$

From the convexity of Ψ', we have a supporting hyperplane at $\boldsymbol{x}_{k+1}$, which satisfies

$$
\begin{aligned}
\Psi'(\boldsymbol{x}) &\geq \Psi'(\boldsymbol{x}_{k+1}) + \langle \boldsymbol{\xi}, \boldsymbol{x}-\boldsymbol{x}_{k+1}\rangle \\
&= \Psi'(\boldsymbol{x}_{k+1}) + \langle g_{L'_{k+1}}(\boldsymbol{y}_{k+1}) - \nabla f'(\boldsymbol{y}_{k+1}), \boldsymbol{x}-\boldsymbol{x}_{k+1}\rangle,
\end{aligned} \tag{48}
$$

for all $\boldsymbol{x} \in \mathbb{R}^n$ and $k \geq 0$. By adding together (47), (48), and the descent condition for f' in (15), we obtain the desired result.

Appendix D
Proof of Theorem 1

All the definitions and results within the scope of this proof hold for all $k \geq 0$. Let the residual describing the tightness of the lower bound w_{k+1} on the objective F at $\boldsymbol{x} \in \mathbb{R}^n$ be denoted by

$$
\begin{aligned}
R_{k+1}(\boldsymbol{x}) &\triangleq F(\boldsymbol{x}) - F(\boldsymbol{x}_{k+1}) - \frac{1}{2L'_{k+1}}\|g_{L'_{k+1}}(\boldsymbol{y}_{k+1})\|_2^2 \\
&\quad - \langle g_{L'_{k+1}}(\boldsymbol{y}_{k+1}), \boldsymbol{x}-\boldsymbol{y}_{k+1}\rangle - \frac{\mu}{2}\|\boldsymbol{x}-\boldsymbol{y}_{k+1}\|_2^2.
\end{aligned} \tag{49}
$$

We introduce the reduced composite gradient $\boldsymbol{G}_{k+1}$ in the form of

$$
\boldsymbol{G}_{k+1} \triangleq g_{L'_{k+1}}(\boldsymbol{y}_{k+1}) - \mu \boldsymbol{y}_{k+1}. \tag{50}
$$

The reduced composite gradient simplifies the non-constant polynomial term in residual expression (49) as

$$
\begin{aligned}
&\langle g_{L'_{k+1}}(\boldsymbol{y}_{k+1}), \boldsymbol{x}-\boldsymbol{y}_{k+1}\rangle + \frac{\mu}{2}\|\boldsymbol{x}-\boldsymbol{y}_{k+1}\|_2^2 \\
&\quad = \langle \boldsymbol{G}_{k+1}, \boldsymbol{x}-\boldsymbol{y}_{k+1}\rangle + \frac{\mu}{2}\|\boldsymbol{x}\|_2^2 - \frac{\mu}{2}\|\boldsymbol{y}_{k+1}\|_2^2.
\end{aligned} \tag{51}
$$

Lemma 1 ensures that $R_{k+1}(\boldsymbol{x}) \geq 0$ for all $\boldsymbol{x} \in \mathbb{R}^n$. Therefore

$$
A_k R_{k+1}(\boldsymbol{x}_k) + a_{k+1} R_{k+1}(\boldsymbol{x}^*) \geq 0. \tag{52}
$$

By expanding (52) using (49) and (51), we obtain that

$$
A_k(F(\boldsymbol{x}_k) - F(\boldsymbol{x}^*)) - A_{k+1}(F(\boldsymbol{x}_{k+1}) - F(\boldsymbol{x}^*)) \geq \mathcal{C}_{k+1},
$$

where the lower bound $\mathcal{C}_{k+1}$ is defined as

$$
\begin{aligned}
\mathcal{C}_{k+1} &\triangleq \mathcal{C}_{k+1}^{(1)} + \langle \boldsymbol{G}_{k+1}, A_k \boldsymbol{x}_k + a_{k+1}\boldsymbol{x}^* - A_{k+1}\boldsymbol{y}_{k+1}\rangle \\
&\quad + \frac{A_k \mu}{2}\|\boldsymbol{x}_k\|_2^2 + \frac{a_{k+1}\mu}{2}\|\boldsymbol{x}^*\|_2^2 - \frac{A_{k+1}\mu}{2}\|\boldsymbol{y}_{k+1}\|_2^2,
\end{aligned} \tag{53}
$$

with

$$
\mathcal{C}_{k+1}^{(1)} \triangleq \frac{A_{k+1}}{2L'_{k+1}}\|g_{L'_{k+1}}(\boldsymbol{y}_{k+1})\|_2^2.
$$

Using the reduced composite gradient definition in (50), we expand $\mathcal{C}_{k+1}^{(1)}$ as

$$
\begin{aligned}
\mathcal{C}_{k+1}^{(1)} &= \mathcal{A}_{k+1} + \frac{a_{k+1}^2}{2\gamma_{k+1}}\|\boldsymbol{G}_{k+1} + \mu \boldsymbol{y}_{k+1}\|_2^2 \\
&= \mathcal{A}_{k+1} + \mathcal{C}_{k+1}^{(2)} + \frac{a_{k+1}^2 \mu}{\gamma_{k+1}}\langle \boldsymbol{G}_{k+1}, \boldsymbol{y}_{k+1}\rangle + \frac{a_{k+1}^2 \mu^2}{2\gamma_{k+1}}\|\boldsymbol{y}_{k+1}\|_2^2,
\end{aligned} \tag{54}
$$

where

$$
\mathcal{C}_{k+1}^{(2)} \triangleq \frac{a_{k+1}^2}{2\gamma_{k+1}}\|\boldsymbol{G}_{k+1}\|_2^2. \tag{55}
$$

Applying (50) in vertex update (27) yields

$$
a_{k+1}\boldsymbol{G}_{k+1} = \gamma_k \boldsymbol{v}_k - \gamma_{k+1}\boldsymbol{v}_{k+1}. \tag{56}
$$

Using (26) and (56) in $\mathcal{C}_{k+1}^{(2)}$ expression (55) we obtain that

$$
\begin{aligned}
\mathcal{C}_{k+1}^{(2)} &= \frac{1}{2\gamma_{k+1}}\|\gamma_k \boldsymbol{v}_k - \gamma_{k+1}\boldsymbol{v}_{k+1}\|_2^2 \\
&= \frac{\gamma_{k+1}}{2}\|\boldsymbol{v}_{k+1}\|_2^2 - \frac{\gamma_k}{2}\|\boldsymbol{v}_k\|_2^2 + \frac{\mu}{2\gamma_{k+1}}a_{k+1}\gamma_k\|\boldsymbol{v}_k\|_2^2 \\
&\quad + \frac{1}{\gamma_{k+1}}\langle \boldsymbol{G}_{k+1}, a_{k+1}\gamma_k \boldsymbol{v}_k\rangle
\end{aligned} \tag{57}
$$

The coefficients of the $\boldsymbol{y}_{k+1}$ terms in $\mathcal{C}_{k+1}$ are given by

$$
A_{k+1}\gamma_{k+1} - a_{k+1}^2\mu = A_k\gamma_{k+1} + a_{k+1}\gamma_k = Y_{k+1}. \tag{58}
$$

Combining (54) and (57) in (53), rearranging terms, and applying (58) yields

$$
\mathcal{C}_{k+1} = \mathcal{A}_{k+1} + V_{k+1} + \frac{1}{\gamma_{k+1}}\langle \boldsymbol{G}_{k+1}, \boldsymbol{s}_{k+1}\rangle + \frac{\mu}{2\gamma_{k+1}}S_{k+1}, \tag{59}
$$

where S_{k+1} and V_{k+1} are, respectively, defined as

$$
\begin{aligned}
S_{k+1} &\triangleq A_k\gamma_{k+1}\|\boldsymbol{x}_k\|_2^2 + a_{k+1}\gamma_k\|\boldsymbol{v}_k\|_2^2 - Y_{k+1}\|\boldsymbol{y}_{k+1}\|_2^2, \\
V_{k+1} &\triangleq \frac{\gamma_{k+1}}{2}\|\boldsymbol{v}_{k+1}\|_2^2 - \frac{\gamma_k}{2}\|\boldsymbol{v}_k\|_2^2 + \langle \boldsymbol{G}_{k+1}, a_{k+1}\boldsymbol{x}^*\rangle \\
&\quad + \frac{a_{k+1}\mu}{2}\|\boldsymbol{x}^*\|_2^2.
\end{aligned} \tag{60}
$$

Applying (26) and (56) in (60) yields

$$
V_{k+1} = \frac{\gamma_{k+1}}{2}\|\boldsymbol{v}_{k+1} - \boldsymbol{x}^*\|_2^2 - \frac{\gamma_k}{2}\|\boldsymbol{v}_k - \boldsymbol{x}^*\|_2^2. \tag{61}
$$

Putting together (53), (59), and (61) we obtain

$$
\Delta_{k+1} + \mathcal{A}_{k+1} + \frac{1}{\gamma_{k+1}}\langle \boldsymbol{G}_{k+1}, \boldsymbol{s}_{k+1}\rangle + \frac{\mu}{2\gamma_{k+1}}S_{k+1} \leq \Delta_k. \tag{62}
$$

For brevity, we define ω_{k+1} as

$$\omega_{k+1} \triangleq \frac{a_{k+1}\gamma_k}{Y_{k+1}}.$$

Residuals $\boldsymbol{s}_{k+1}$ and S_{k+1} can thus be written as

$$\boldsymbol{s}_{k+1} = Y_{k+1}((1-\omega_{k+1})\boldsymbol{x}_k + \omega_{k+1}\boldsymbol{v}_k - \boldsymbol{y}_{k+1}), \tag{63}$$

$$S_{k+1} = Y_{k+1}\left((1-\omega_{k+1})\|\boldsymbol{x}_k\|_2^2 + \omega_{k+1}\|\boldsymbol{v}_k\|_2^2 - \|\boldsymbol{y}_{k+1}\|_2^2\right). \tag{64}$$

Residual S_{k+1} can be expressed in terms of $\boldsymbol{s}_{k+1}$ using the following identity:

$$(1-\omega_{k+1})\|\boldsymbol{x}_k\|_2^2 + \omega_{k+1}\|\boldsymbol{v}_k\|_2^2 = ((1-\omega_{k+1})\boldsymbol{x}_k + \omega_{k+1}\boldsymbol{v}_k)^2 + (1-\omega_{k+1})\omega_{k+1}\|\boldsymbol{x}_k - \boldsymbol{v}_k\|_2^2. \tag{65}$$

The proof of (65) is obtained simply by rearranging terms. Using (63) and (65) in (64), we obtain that

$$S_{k+1} = Y_{k+1}\left(((1-\omega_{k+1})\boldsymbol{x}_k + \omega_{k+1}\boldsymbol{v}_k)^2 - \|\boldsymbol{y}_{k+1}\|_2^2\right) + S_{k+1}^{(1)} = \left\langle \frac{1}{Y_{k+1}}\boldsymbol{s}_{k+1} + 2\boldsymbol{y}_{k+1}, \boldsymbol{s}_{k+1}\right\rangle + S_{k+1}^{(1)}, \tag{66}$$

where $S_{k+1}^{(1)}$ is defined as

$$S_{k+1}^{(1)} \triangleq Y_{k+1}(1-\omega_{k+1})\omega_{k+1}\|\boldsymbol{x}_k - \boldsymbol{v}_k\|_2^2 = \frac{a_{k+1}A_k\gamma_k\gamma_{k+1}}{A_k\gamma_{k+1} + a_{k+1}\gamma_k}\|\boldsymbol{x}_k - \boldsymbol{v}_k\|_2^2.$$

The square term $\|\boldsymbol{x}_k - \boldsymbol{v}_k\|_2^2$ is always non-negative, hence

$$S_{k+1}^{(1)} \geq 0. \tag{67}$$

Putting together (50), (66), and (67) yields

$$\begin{aligned}\frac{1}{\gamma_{k+1}}\langle \boldsymbol{G}_{k+1}, \boldsymbol{s}_{k+1}\rangle + \frac{\mu}{2\gamma_{k+1}}S_{k+1} &\geq \frac{1}{\gamma_{k+1}}\left\langle \boldsymbol{G}_{k+1} + \frac{\mu}{2}\left(\frac{1}{Y_{k+1}}\boldsymbol{s}_{k+1} + 2\boldsymbol{y}_{k+1}\right), \boldsymbol{s}_{k+1}\right\rangle \\ &= \frac{1}{\gamma_{k+1}}\left\langle g_{L'_{k+1}}(\boldsymbol{y}_{k+1}) + \frac{\mu}{2Y_{k+1}}\boldsymbol{s}_{k+1}, \boldsymbol{s}_{k+1}\right\rangle.\end{aligned} \tag{68}$$

Combining (62) with (68) gives the desired result.

Appendix E
Proof of Theorem 2

In the non-strongly convex case, we have

$$A_{k+1} = A_k + a_{k+1} \overset{(28)}{\geq} A_k + \frac{1+\sqrt{1+4L_{k+1}A_k}}{2L_{k+1}} \overset{(33)}{\geq} A_k + \frac{1}{2L_u} + \sqrt{\frac{1}{4L_u^2} + \frac{A_k}{L_u}}, \quad k \geq 0. \tag{69}$$

We prove by induction that (34) holds for all $k \geq 1$. First, for $k = 1$, (34) is valid since

$$A_1 = \frac{1}{L_1} \geq \frac{(1+1)^2}{4L_u}.$$

Next, we assume that (34) is valid for $k \geq 1$, and show that it holds for $k+1$. From (34) and (69), we have

$$\begin{aligned}A_{k+1} &\geq \frac{(k+1)^2}{4L_u} + \frac{1}{2L_u} + \sqrt{\frac{1}{4(L_u)^2} + \frac{(k+1)^2}{4(L_u)^2}} \\ &= \frac{1}{4L_u}\left((k+1)^2 + 2 + 2\sqrt{1+(k+1)^2}\right) \geq \frac{(k+2)^2}{4L_u}.\end{aligned}$$

In the strongly convex case, the curvature of the estimate function can be expressed in absolute terms as

$$\gamma_k = \gamma_0 + \left(\sum_{i=1}^{k} a_i\right)\mu = \gamma_0 + (A_k - A_0)\mu = 1 + A_k\mu, \ k \geq 0,$$

which trivially implies that $\gamma_k > A_k\mu$. Hence, we have

$$\frac{a_{k+1}^2}{A_{k+1}^2} \overset{(30)}{=} \frac{\gamma_{k+1}}{(L_{k+1}+\mu_\Psi)A_{k+1}} > \frac{\mu}{L_{k+1}+\mu_\Psi} = q_{k+1} \geq q_u,$$

for all $k \geq 0$. This leads to

$$\frac{A_{k+1}}{A_k} > \frac{1}{1-\sqrt{q_u}}, \quad k \geq 1.$$

Using $A_1 = \frac{1}{L_1-\mu_f} \geq \frac{1}{L_u-\mu_f}$, the rate lower bound in (35) follows by induction.

Appendix F
Proof of Lemma 2

By combining (29) with (32), we get

$$\begin{aligned}\boldsymbol{v}_{k+1} &= \frac{\gamma_k}{\gamma_{k+1}}\frac{(a_{k+1}\gamma_k + A_k\gamma_{k+1})\boldsymbol{y}_{k+1} - A_k\gamma_{k+1}\boldsymbol{x}_k}{a_{k+1}\gamma_k} \\ &\quad + \frac{a_{k+1}(L_{k+1}+\mu_\Psi)}{\gamma_{k+1}}\boldsymbol{x}_{k+1} - \frac{a_{k+1}(L_{k+1}-\mu_f)}{\gamma_{k+1}}\boldsymbol{y}_{k+1} \\ &\overset{(30)}{=} \frac{a_{k+1}\gamma_k + A_k\gamma_{k+1} - A_{k+1}\gamma_{k+1} - a_{k+1}^2\mu}{a_{k+1}\gamma_{k+1}}\boldsymbol{y}_{k+1} \\ &\quad + \frac{A_{k+1}}{a_{k+1}}\boldsymbol{x}_{k+1} - \frac{A_k}{a_{k+1}}\boldsymbol{x}_k \\ &\overset{(26)}{=} \boldsymbol{x}_k + \frac{A_{k+1}}{a_{k+1}}(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k), \quad k \geq 0.\end{aligned}$$

Appendix G
Proof of Proposition 2

With judicious use of parameters r_u and r_d, the average WTU cost of an ACGM iteration can be adjusted to equal that of FISTA (also evidenced in Subsection V-A). Consequently, it is adequate to compare the convergence guarantees of the two algorithms when indexed in iterations.

Combining (39) and (42), we obtain for all $k \geq 0$ that

$$A_{k+1}^{\mathrm{ACGM}} = \left(\sqrt{\frac{1}{4L_{k+1}^{\mathrm{ACGM}}}} + \sqrt{\frac{1}{4L_{k+1}^{\mathrm{ACGM}}} + A_k^{\mathrm{ACGM}}}\right)^2.$$

Replacing (42) in ACGM with

$$t_{k+1}^{\mathrm{FISTA}} = \frac{1+\sqrt{1+4\left(t_k^{\mathrm{FISTA}}\right)^2}}{2}, \quad k \geq 0, \tag{70}$$

results in an algorithm that produces identical iterates to FISTA. The convergence analysis of ACGM employing (70) instead of (42) yields for all $k \geq 0$ the following expression:

$$A_{k+1}^{\mathrm{FISTA}} = \left(\sqrt{\frac{1}{4L_{k+1}^{\mathrm{FISTA}}}} + \sqrt{\frac{1}{4L_{k+1}^{\mathrm{FISTA}}} + \frac{L_k^{\mathrm{FISTA}}}{L_{k+1}^{\mathrm{FISTA}}} A_k^{\mathrm{FISTA}}} \right)^2 .$$

Both methods start with the same state, in which we have $A_0^{\mathrm{ACGM}} = A_0^{\mathrm{FISTA}} = 0$. The line-search procedure of ACGM is guaranteed to produce Lipschitz constant estimates no greater than those of FISTA for the same local curvature, i.e., $L_k^{\mathrm{ACGM}} \leq L_k^{\mathrm{FISTA}}$, $k \geq 0$. FISTA, by design, can only accommodate a Lipschitz constant estimate increase, namely $L_k^{\mathrm{FISTA}} \leq L_{k+1}^{\mathrm{FISTA}}$, $k \geq 0$. Thus, for any variation in the local curvature of f, we have

$$A_k^{\mathrm{ACGM}} \geq A_k^{\mathrm{FISTA}}, \quad k \geq 0.$$

REFERENCES

[1] P. L. Combettes and J.-C. Pesquet, "Proximal splitting methods in signal processing," in *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*. Berlin, Germany: Springer, 2011, pp. 185–212.

[2] K. Slavakis, G. B. Giannakis, and G. Mateos, "Modeling and optimization for big data analytics: (Statistical) learning tools for our era of data deluge," *IEEE Signal Process. Mag.*, vol. 31, no. 5, pp. 18–31, Sep. 2014.

[3] V. Cevher, S. Becker, and M. Schmidt, "Convex optimization for big data: Scalable, randomized, and parallel algorithms for big data analytics," *IEEE Signal Process. Mag.*, vol. 31, no. 5, pp. 32–43, Sep. 2014.

[4] Y. Nesterov, "Subgradient methods for huge-scale optimization problems," *Math. Program., Ser. A*, vol. 146, no. 1/2, pp. 275–297, 2014.

[5] Y. Nesterov, "A method of solving a convex programming problem with convergence rate $\mathcal{O}(1/k^2)$," *Dokl. Math.*, vol. 27, no. 2, pp. 372–376, 1983.

[6] Y. Nesterov, *Introductory Lectures on Convex Optimization. Applied Optimization*, vol. 87. Boston, MA, USA: Kluwer, 2004.

[7] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imag. Sci.*, vol. 2, no. 1, pp. 183–202, 2009.

[8] A. Chambolle and T. Pock, "An introduction to continuous optimization for imaging," *Acta Numer.*, vol. 25, pp. 161–319, 2016.

[9] M. Baes, "Estimate sequence methods: Extensions and approximations," May 2017. [Online]. Available: http://www.optimization-online.org/DB_FILE/2009/08/2372.pdf

[10] W. W. Hager, D. T. Phan, and H. Zhang, "Gradient-based methods for sparse recovery," *SIAM J. Imag. Sci.*, vol. 4, no. 1, pp. 146–165, 2011.

[11] Z. Wen, W. Yin, D. Goldfarb, and Y. Zhang, "A fast algorithm for sparse reconstruction based on shrinkage, subspace optimization, and continuation," *SIAM J. Sci. Comput.*, vol. 32, no. 4, pp. 1832–1857, 2010.

[12] Z. Wen, W. Yin, H. Zhang, and D. Goldfarb, "On the convergence of an active-set method for l_1 minimization," *Optim. Methods Softw.*, vol. 27, no. 6, pp. 1127–1146, 2012.

[13] S. J. Wright, R. D. Nowak, and M. A. Figueiredo, "Sparse reconstruction by separable approximation," *IEEE Trans. Signal Process.*, vol. 57, no. 7, pp. 2479–2493, Jul. 2009.

[14] A. Nemirovski and D.-B. Yudin, *Problem Complexity and Method Efficiency in Optimization*. New York, NY, USA: Wiley, 1983.

[15] Y. Nesterov, "Gradient methods for minimizing composite functions," *Math. Program. B*, vol. 140, no. 1, pp. 125–161, 2013.

[16] N. Parikh *et al.*, "Proximal algorithms," *Found. Trends Optim.*, vol. 1, no. 3, pp. 127–239, 2014.

[17] P. Tseng, "On accelerated proximal gradient methods for convex-concave optimization," *SIAM J. Optim.*, submitted to be published, 2008.

[18] R. D. C. Monteiro, C. Ortiz, and B. F. Svaiter, "An adaptive accelerated first-order method for convex optimization," *Comput. Optim. Appl.*, vol. 64, no. 1, pp. 31–73, 2016.

[19] Q. Lin and L. Xiao, "An adaptive accelerated proximal gradient method and its homotopy continuation for sparse optimization," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 73–81.

[20] A. Chambolle and C. Dossal, "On the convergence of the iterates of the 'Fast iterative shrinkage/thresholding algorithm'," *J. Optim. Theory Appl.*, vol. 166, no. 3, pp. 968–982, 2015.

[21] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.

[22] L. Armijo, "Minimization of functions having Lipschitz continuous first partial derivatives," *Pacific J. Math*, vol. 16, no. 1, pp. 1–3, 1966.

[23] S. R. Becker, E. J. Candès, and M. C. Grant, "Templates for convex cone problems with applications to sparse signal recovery," *Math. Program. Comput.*, vol. 3, no. 3, pp. 165–218, 2011.

[24] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, 5th ed. San Francisco, CA, USA: Morgan Kaufmann Publishers, 2011.

[25] Y. Nesterov, "Efficiency of coordinate descent methods on huge-scale optimization problems," *SIAM J. Optim.*, vol. 22, no. 2, pp. 341–362, 2012.

[26] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. Roy. Statist. Soc. B. Methodol.*, vol. 58, no. 1, pp. 267–288, 1996.

[27] M. I. Florea and S. A. Vorobyov, "A robust FISTA-like algorithm," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Mar. 2017, New Orleans, LA, USA, pp. 4521–4525.

[28] P. C. Hansen, J. G. Nagy, and D. P. O'Leary, *Deblurring Images: Matrices, Spectra, and Filtering*. Philadelphia, PA, USA: SIAM, 2006.

[29] D. R. Cox, "The regression analysis of binary sequences," *J. Roy. Statist. Soc. B. Methodol.*, vol. 20, pp. 215–242, 1958.

[30] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *J. Roy. Statist. Soc. B. Methodol.*, vol. 67, no. 2, pp. 301–320, 2005.

[31] B. O'Donoghue and E. Candès, "Adaptive restart for accelerated gradient schemes," *Found. Comput. Math.*, vol. 15, no. 3, pp. 715–732, 2015.

[32] K. Scheinberg, D. Goldfarb, and X. Bai, "Fast first-order methods for composite convex optimization with backtracking," *Found. Comput. Math.*, vol. 14, no. 3, pp. 389–417, 2014.

Mihai I. Florea (S'17) received the B.E. degree in communication systems from Tohoku University, Sendai, Japan, in 2009, and the M.Sc. degree in computer science from Uppsala University, Uppsala, Sweden, in 2013. He is currently working toward the D.Sc. degree in signal processing technology at Aalto University, Aalto, Finland. From 2004 to 2009, he was a recipient of the Japanese Government (MEXT) Scholarship and in 2009 was awarded the Tohoku University Head of School of Engineering Prize. His research interests include convex optimization algorithms and inverse problems in medical imaging.

Sergiy A. Vorobyov (M'02–SM'05–F'18) is currently a Professor with the Department of Signal Processing and Acoustics, Aalto University, Aalto, Finland. He was previously with the University of Alberta, Alberta, Canada as an Assistant, Associate, and then Full Professor. Since his graduation, he also held various research and faculty positions at Kharkiv National University of Radio Electronics, Ukraine; the Institute of Physical and Chemical Research (RIKEN), Japan; McMaster University, Canada; Duisburg-Essen University and Darmstadt University of Technology, Germany; and Heriot-Watt University, U.K. His research interests include optimization and multi-liner algebra methods and algorithms with applications in signal processing; statistical and array signal processing; sparse signal processing; estimation and detection theory; sampling theory; and multi-antenna, large-scale, cooperative, and cognitive systems and algorithms. He is a recipient of the 2004 IEEE Signal Processing Society Best Paper Award, the 2007 Alberta Ingenuity New Faculty Award, the 2011 Carl Zeiss Award (Germany), the 2012 NSERC Discovery Accelerator Award, and other awards.