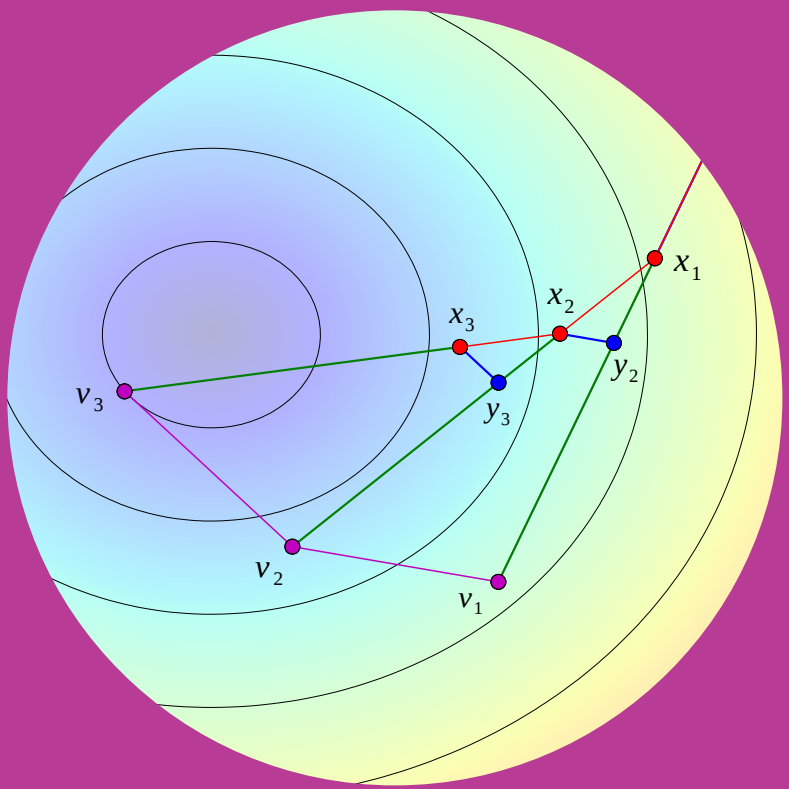
Department of Signal Processing and Acoustics

Constructing Accelerated Algorithms for Large-scale Optimization

Framework, Algorithms, and Applications

Mihai Iulian Florea



Aalto University

DOCTORAL DISSERTATIONS

Mihai Iulian Florea Constructing Accelerated Algorithms for Large-scale Optimization

Aalto University

2018

Aalto University publication series
DOCTORAL DISSERTATIONS 197/2018

Constructing Accelerated Algorithms for Large-scale Optimization

Framework, Algorithms, and Applications

Mihai Iulian Florea

A doctoral dissertation completed for the degree of Doctor of Science (Technology) to be defended, with the permission of the Aalto University School of Electrical Engineering, at a public examination held at the lecture hall TU2 of the school on 23 October 2018 at 12:00.

Aalto University
School of Electrical Engineering
Department of Signal Processing and Acoustics
Signal Processing Group

Supervising professor
Professor Sergiy A. Vorobyov, Aalto University, Finland

Preliminary examiners
Professor Yurii Nesterov, Catholic University of Louvain, Belgium
Professor José M. Bioucas Dias, University of Lisbon, Portugal

Opponents
Professor Yurii Nesterov, Catholic University of Louvain, Belgium
Professor Jean-Christophe Pesquet, University of Paris-Saclay, France

Aalto University publication series
DOCTORAL DISSERTATIONS 197/2018

ISBN 978-952-60-8226-4 (printed)
ISBN 978-952-60-8227-1 (pdf)
ISSN 1799-4934 (printed)
ISSN 1799-4942 (pdf)
http://urn.fi/URN:ISBN:978-952-60-8227-1

Unigrafia Oy
Helsinki 2018

Finland

441 697
Printed matter

Aalto University

Abstract

Aalto University, P.O. Box 11000, FI-00076 Aalto www.aalto.fi

Author
Mihai Iulian Florea

Name of the doctoral dissertation
Constructing Accelerated Algorithms for Large-scale Optimization

Publisher School of Electrical Engineering

Unit Department of Signal Processing and Acoustics

Series Aalto University publication series DOCTORAL DISSERTATIONS 197/2018

Field of research Signal Processing Technology

Manuscript submitted 11 June 2018 **Date of the defence** 23 October 2018

Permission to publish granted (date) 30 August 2018 **Language** English

☐ **Monograph** ☒ **Article dissertation** ☐ **Essay dissertation**

Abstract

A wide range of inverse problems and various machine learning tasks can be expressed as large-scale optimization problems with a composite objective structure, where the gradient of the smooth part has a global Lipschitz constant that is either impractical to compute or considerably larger than the local values. The smooth part may be strongly convex as well, especially in certain medical imaging applications. The only fast methods that are able to address this entire class of problems are similar to or based on the black-box algorithms developed and analyzed by Nesterov using the estimate sequence mathematical framework.

In this work, we introduce 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 the entire composite problem class. ACGM is endowed with an efficient dynamic Lipschitz constant estimation (line-search) procedure and features monotonicity.

Motivated by the absence of an accurate complexity measure applicable to all first-order methods, we also introduce the wall-clock time unit (WTU). The WTU accounts for variations in algorithmic per-iteration complexity and more consistently reflects the performance of first-order methods in practical implementations. 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. We confirm the superiority of ACGM within its class using an extensive simulation benchmark.

ACGM also excels in terms of robustness and usability. In particular, ACGM is guaranteed to converge without requiring any quantitative prior information on the problem. Additional information, if available, leads to an improvement in performance at least on par with competing methods. Moreover, ACGM actually encompasses several popular algorithms for large-scale optimization, including Nesterov's Fast Gradient Method (FGM) and the Fast Iterative Shrinkage Thresholding Algorithm (FISTA), along with their most common variants.

The efficiency and generality of ACGM enables new applications, particularly in ultrasound image reconstruction. In contrast with the unrealistic models of existing approaches, we propose two ultrasound image formation models based on spatially varying kernel convolution that account for arbitrary boundary conditions. We provide these models and their adjoints with resource efficient matrix-free implementations. Using either of our models, a variant of ACGM optimized for this task is able to efficiently reconstruct large ultrasound images with accuracy vastly superior to the state-of-the-art.

Keywords Acceleration, composite objective, estimate sequence, Fast Gradient Method, first-order method, FISTA, large-scale optimization, line-search, Nesterov method, matrix-free, optimization algorithm, point-spread function, spatially varying, ultrasound

ISBN (printed) 978-952-60-8226-4 **ISBN (pdf)** 978-952-60-8227-1

ISSN (printed) 1799-4934 **ISSN (pdf)** 1799-4942

Location of publisher Helsinki **Location of printing** Helsinki **Year** 2018

Pages 203 **urn** http://urn.fi/URN:ISBN:978-952-60-8227-1

Preface

The research work that has led to this doctoral thesis has been carried out in the Department of Signal Processing and Acoustics at Aalto University, Finland. The research on the topic of ultrasound image reconstruction was conducted in collaboration with the IRIT UMR CNRS 5505 Laboratory at Université Paul Sabatier Toulouse 3, France.

This work has been partially supported by the Academy of Finland under Grant 299243, the Aalto ELEC Doctoral School, the CIMI Labex, Toulouse, France under Grant ANR-11-LABX-0040-CIMI, and the Erasmus+ Mobility Programme.

I wish to thank all who have contributed, either directly or indirectly, to the completion of this dissertation.

Otaniemi, October 23, 2018,

Mihai Iulian Florea

Contents

Preface **5**

Contents **7**

List of Publications **11**

Author's Contribution **13**

List of Abbreviations **15**

List of Symbols **17**

1. Introduction **23**
- 1.1 Large-scale Optimization 23
- 1.2 Motivation 24
- 1.3 Objectives and Scope 25
- 1.4 Contributions 26
  - 1.4.1 Large-scale Convex Optimization Algorithms . . 26
  - 1.4.2 Ultrasound Image Reconstruction 27
- 1.5 Author's Independent Contribution 28
- 1.6 Thesis Structure 28

2. Augmenting the Estimate Sequence **31**
- 2.1 The Large-scale Composite Problem Class 31
- 2.2 Complexity Bounds 32
- 2.3 Convergence Guarantees 33
- 2.4 The Estimate Sequence 35
  - 2.4.1 A Substitute Convergence Guarantee 35
  - 2.4.2 Nesterov's Estimate Sequences 36
- 2.5 The Augmented Estimate Sequence 39
- 2.6 Parabolae 40
  - 2.6.1 Parabolic Estimate Functions 41
  - 2.6.2 Composite Parabolae 42

2.7 The Gap Sequence 42

3. Constructing ACGM **45**
3.1 A Design Pattern for First-order Accelerated Algorithms . 45
3.1.1 Line-search 48
3.2 Design Choices 48
3.2.1 Upper Bounds 49
3.3 ACGM for Non-strongly Convex Objectives 51
3.3.1 Lower Bounds 51
3.3.2 Formulating ACGM 52
3.3.3 Extrapolated Form 56
3.4 ACGM for Objectives with Arbitrary Strong Convexity . . 59
3.4.1 Strong Convexity Transfer 59
3.4.2 Lower Bounds 61
3.4.3 Generalizing ACGM to Arbitrary Strong Convexity 62
3.5 Monotone ACGM 74
3.5.1 Upper Bounds 74
3.5.2 Formulating Monotone ACGM 74
3.5.3 Extrapolated Form 77

4. Analysis of ACGM **83**
4.1 Revisiting the Estimate Sequence 83
4.2 Worst-case Convergence Guarantees 86
4.3 Wall-clock Time Units 89
4.3.1 Standard WTU 90
4.3.2 Generalized WTU 93
4.4 ACGM among its Class of Algorithms 95
4.4.1 Uniting Nesterov's FGM and FISTA 95
4.4.2 Standard WTU Worst-case Analysis 96
4.4.3 Theoretical Superiority of ACGM 99

5. Simulations **103**
5.1 Non-monotone ACGM Benchmark 103
5.1.1 l_1-regularized Image Deblurring 103
5.1.2 Logistic Regression with Elastic Net 106
5.2 Monotone ACGM Benchmark 109
5.2.1 Benchmark Setup 109
5.2.2 Non-strongly Convex Problems 111
5.2.3 Strongly Convex Problems 114

6. Ultrasound Image Reconstruction **117**
6.1 Background 117
6.1.1 Pulse-echo Ultrasound 118
6.1.2 Previous Work 118
6.2 Notation 119

6.3 Discrete Convolution 120
6.3.1 Definitions 120
6.3.2 Adjoint Expressions 122
6.4 Ultrasound Image Formation Models 125
6.4.1 Prototype Mixture Model 125
6.4.2 Axially Variant Kernel Model 128
6.5 Optimizing ACGM for Linear Inverse Problems 130
6.6 Experimental Results 131
6.6.1 Prototype Mixture Model 131
6.6.2 Axially Variant Kernel Model 135

7. Conclusions **139**

References **141**

Publications **147**

List of Publications

This thesis consists of an overview and of the following publications which are referred to in the text by their Roman numerals.

I Mihai I. Florea and Sergiy A. Vorobyov. A Robust FISTA-like Algorithm. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, New Orleans, USA, pp. 4521–4525, Mar. 2017.

II Mihai I. Florea and Sergiy A. Vorobyov. An Accelerated Composite Gradient Method for Large-scale Composite Objective Problems. Accepted for publication in *IEEE Transactions on Signal Processing*, May 2018.

III Mihai I. Florea and Sergiy A. Vorobyov. A Generalized Accelerated Composite Gradient Method: Uniting Nesterov's Fast Gradient Method and FISTA. Submitted to *IEEE Transactions on Signal Processing*, Oct. 2018.

IV Mihai I. Florea, Adrian Basarab, Denis Kouamé, and Sergiy A. Vorobyov. Restoration of Ultrasound Images using Spatially-variant Kernel Deconvolution. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, Calgary, Canada, pp. 796–800, Apr. 2018.

V Mihai I. Florea, Adrian Basarab, Denis Kouamé, and Sergiy A. Vorobyov. An Axially Variant Kernel Imaging Model Applied to Ultrasound Image Reconstruction. *IEEE Signal Processing Letters*, vol. 25, no.7, pp. 961–965, Jul. 2018.

Author's Contribution

Publication I: "A Robust FISTA-like Algorithm"

The main author proposed the idea, derived the theoretical results, and performed the simulations with input from the co-author.

Publication II: "An Accelerated Composite Gradient Method for Large-scale Composite Objective Problems"

The main author proposed the idea, derived the theoretical results, and performed the simulations with input from the co-author.

Publication III: "A Generalized Accelerated Composite Gradient Method: Uniting Nesterov's Fast Gradient Method and FISTA"

The main author proposed the idea, derived the theoretical results, and performed the simulations with input from the co-author.

Publication IV: "Restoration of Ultrasound Images using Spatially-variant Kernel Deconvolution"

The main author proposed the idea, derived the theoretical results, and performed the simulations with input from the co-authors.

Publication V: "An Axially Variant Kernel Imaging Model Applied to Ultrasound Image Reconstruction"

The main author proposed the idea, derived the theoretical results, and performed the simulations with input from the co-authors.

List of Abbreviations

1D	1 Dimensional
2D	2 Dimensional
3D	3 Dimensional
4D	4 Dimensional
AA	Adaptive Accelerated (method)
ACGM	Accelerated Composite Gradient Method
AESP	Augmented Estimate Sequence Property
AI	Axially Invariant (deconvolution result)
AMGS	Accelerated Multistep Gradient Scheme
AV	Axially Variant (deconvolution result)
AWGN	Additive White Gaussian Noise
B-mode	Brightness mode
BACGM	Border-case ACGM
BMACGM	Border-case Monotone ACGM
CDM	Coordinate Descent Method
CPU	Central Processing Unit
dB	decibel
DFT	Discrete Fourier Transform
EN	Elastic Net (problem)
ES	Estimate Sequence
ESP	Estimate Sequence Property
FGM	Fast Gradient Method
FISTA	Fast Iterative Shrinkage-Thresholding Algorithm
FISTA-BT	FISTA with BackTracking (line-search)
FISTA-CP	FISTA Chambolle-Pock
GPU	Graphics Processing Unit
i.i.d.	independent identically distributed
ISD	Image Space Distance
ISDUB	Image Space Distance Upper Bound
L1LR	l_1-regularized Logistic Regression
LASSO	Least Absolute Shrinkage and Selection Operator (problem)
LCE	Lipschitz Constant Estimate

LSSC	Line-Search Stopping Criterion
MACGM	Monotone ACGM
MC	Monotone Condition
MFISTA	Monotone FISTA
MFISTA-CP	Monotone FISTA-CP
MHz	megahertz
mm	milimeter
MOS	Monteiro-Ortiz-Svaiter (method)
NNLS	Non-Negative Least Squares
PPU	Parallel Processing Unit
PSF	Point-Spread Function
RF	Radio-Frequency
RHS	Right-Hand Side
RR	Ridge Regression (problem)
scAPG	strongly convex Accelerated Proximal Gradient (method)
SD	Standard Deviation
TRF	Tissue Reflectivity Function
UMA	Uniform Memory Access
WTU	Wall-clock Time Unit

List of Symbols

$\mathbf{1}_X$	Membership function of set X
$\boldsymbol{A}$	Real valued matrix, often a model matrix, that is multiplied with $\boldsymbol{x}$ in a number of optimization problems
$\boldsymbol{a}$	2D image
$\bar{\boldsymbol{a}}$	2D image padded with zeros
A_k	Convergence guarantee ($k \geq 0$)
$\tilde{A}_k$	Scaled A_k
$\mathcal{A}_{k+1}$	Subexpression used in Theorem 4 ($k \geq 0$)
a_{k+1}	Estimate sequence weight and convergence guarantee increment ($k \geq 0$)
b_k	Auxiliary point extrapolation factor if the monotone condition passes ($k \geq 0$)
b'_k	Auxiliary point extrapolation factor if the monotone condition fails ($k \geq 0$)
$\mathcal{B}_{k+1}$	Subexpression used in Theorem 4 ($k \geq 0$)
B^{X}	Asymptotic convergence rate of Algorithm X expressed in WTU
$\mathcal{C}$	Process that computes the new parameters from the old state. Subscripts denote the output parameters.
$\boldsymbol{\mathcal{C}}_1(\boldsymbol{k})$	Discrete full convolution with kernel $\boldsymbol{k}$ operator
$\boldsymbol{\mathcal{C}}_2(\boldsymbol{k})$	Discrete valid convolution with kernel $\boldsymbol{k}$ operator
$\boldsymbol{\mathcal{C}}(\bar{\boldsymbol{k}})$	Discrete circular convolution with kernel $\bar{\boldsymbol{k}}$ operator
$\mathcal{C}_{k+1}$	Subexpression used in Theorem 4 ($k \geq 0$)
$\mathcal{C}_{k+1}^{(1)}$	Subexpression used in Theorem 4 ($k \geq 0$)
$\mathcal{C}_{k+1}^{(2)}$	Subexpression used in Theorem 4 ($k \geq 0$)
c_q	Center row of prototype kernel q
C^{X}	Proportionality constant of the lower bound on the worst-case convergence rate of Algorithm X expressed in WTU
$\mathcal{D}_f$	Set of TRFs
$\boldsymbol{d}_k$	Difference term in Monotone ACGM ($k \geq 0$)
$\mathrm{diag}(\boldsymbol{x})$	Diagonal matrix (linear operator) with the entries of $\boldsymbol{x}$

$\boldsymbol{e}(p, q)$	Standard basis vector image with the value of the pixel at (p, q) equal to 1 and all others equal to 0
$\mathcal{E}(\gamma_k, A_k, L_{k+1})$	Expression of an upper bound on a_{k+1} ($k \geq 0$)
$\boldsymbol{F}$	Discrete Fourier Transform operator
f_0	Ultrasound central frequency in MHz
$\mathcal{F}_{L_f}^{\infty,1}(\mathbb{R}^n)$	Class of problems where the objective is non-strongly convex differentiable with Lipschitz gradient (Lipschitz constant L_f) and the number of variables is n
f_s	Ultrasound sampling frequency in MHz
$F(\boldsymbol{x})$	Composite objective
$f(\boldsymbol{x})$	Smooth part of composite objective F
$f'(\boldsymbol{x})$	Smooth part of composite objective F incorporating all the strong convexity of Ψ
$\mathcal{G}$	Set of all generalized parabolae
$\boldsymbol{G}_{k+1}$	Reduced composite gradient ($k \geq 0$)
$g_L(\boldsymbol{y})$	Composite gradient at point $\boldsymbol{y}$ with step size L
$\mathcal{H}$	Set of all hyperplanes
$\mathbf{H}$	Hessian operator
$\boldsymbol{H}$	Spatially varying kernel convolution operator
H_k	Highest upper bound that can be placed on weighted objective value $A_k F(\boldsymbol{x}_k)$ for $k \geq 0$
$\mathcal{H}_k$	Lower bound on H_k for $k \geq 0$ that is based on W_k for $k \geq 1$
$\mathcal{I}(a, b, c)$	Exception index set containing indices from 1 to c outside the range from a to b
$\mathcal{I}(\boldsymbol{x})$	Sum softplus function
i_h	Output image row
$\boldsymbol{I}_n$	Identity matrix of size n
K	Total number of iterations
k	Current iteration index
$\tilde{k}$	Number of consecutive iterations, starting at $k = 0$, for which no backtracks occur
$\boldsymbol{k}$	Convolution kernel
$\bar{\boldsymbol{k}}$	Convolution kernel padded with zeros
$\boldsymbol{k}(i_h)$	Kernel at row i_h
$\boldsymbol{k}(i, q)$	Kernel at row i next to the qth prototype center
$\boldsymbol{k}_q$	qth prototype kernel
$\mathcal{L}$	Validation rule for Lipschitz constant estimate candidates
L_0	Initial estimate of the Lipschitz constant
L_3	Objective Hessian Lipschitz constant
L_f	Smooth part f gradient Lipschitz constant
$L_{f'}$	f' gradient Lipschitz constant
L_{k+1}	Lipschitz constant estimate for f ($k \geq 0$)
L'_{k+1}	Lipschitz constant estimate for f' ($k \geq 0$)

$l_k(\boldsymbol{x})$	Linear function part of Nesterov's newer estimate sequence variant ($k \geq 0$)
$\tilde{l}_k(\boldsymbol{x})$	Lower linear model on the objective derived from l_k
L_u	Worst-case Lipschitz constant estimate
$\mathcal{L}(\boldsymbol{x})$	Element-wise logistic function
m	Number of rows in matrix $\boldsymbol{A}$
m_1	Placeholder for index 1
m_M	Full convolution of $\boldsymbol{a}$ and $\boldsymbol{k}$ result height
m_N	Valid convolution of $\boldsymbol{a}$ and $\boldsymbol{k}$ result height
m_p	Height of the padded TRF
m_r	Height of the padding boundary / kernel axial radius
m_t	Height of the TRF
n	Number of optimization variables
$\boldsymbol{n}$	Independent identically distributed additive white Gaussian noise
n_1	Placeholder for index 1
n_k	Number of prototype kernels
$\mathcal{N}_{k+1}$	Line search residual ($k \geq 0$)
n_M	Full convolution of $\boldsymbol{a}$ and $\boldsymbol{k}$ result width
n_N	Valid convolution of $\boldsymbol{a}$ and $\boldsymbol{k}$ result width
n_p	Width of the padded TRF
n_r	Width of the padding boundary / kernel lateral radius
n_t	Width of the TRF
$\mathcal{N}(\mu, \sigma^2)$	Standard Gaussian distribution with mean μ and standard deviation σ
$\mathcal{O}(.)$	Order (limiting factor) of the function argument
$\boldsymbol{P}$	Padding operator
$\mathcal{P}$	Set of all parabolae
$P_{f,\boldsymbol{y}}(\boldsymbol{x})$	Abbreviated hyperplane expression at point $\boldsymbol{x}$, parametrized by function f and control point $\boldsymbol{y}$
$\boldsymbol{P}_m$	Operator that pads every column of the input image independently
$\boldsymbol{\mathcal{P}}(m_t, m_r)$	1D padding operator with input size m_t and boundary m_r
$\boldsymbol{P}_n$	Operator that pads every row of the input image independently
$\text{prox}_{\tau\Psi}(\boldsymbol{x})$	Proximal map of regularizer Ψ applied to vector $\boldsymbol{x}$ with step size τ
$\mathcal{P}_\Psi$	Set of all composite parabolae based on Ψ
q	Composite objective inverse condition number
$Q_{f,\gamma,\boldsymbol{y}}(\boldsymbol{x})$	Abbreviated parabola expression at point $\boldsymbol{x}$, parametrized by function f, curvature γ, and control point $\boldsymbol{y}$
q_k	Local inverse condition number ($k \geq 0$)
q_u	Worst-case local inverse condition number
$\mathbb{R}$	Set of real numbers

$\boldsymbol{\mathcal{R}}$	Rotation operator
r_d	Lipschitz constant estimate decrease coefficient
$\mathcal{R}_{f,\Psi,L,\boldsymbol{y}}(\boldsymbol{x})$	Relaxed supporting generalized parabola of objective $F = f + \Psi$ at point $\boldsymbol{y}$ using inverse step size L
R_{k+1}	Residual describing the tightness of lower bound w_{k+1} on objective F ($k \geq 0$)
$\mathbb{R}^n$	Set of n real valued vectors
r_u	Lipschitz constant estimate increase coefficient
S_{k+1}	Subexpression used in Theorem 4 ($k \geq 0$)
$\boldsymbol{s}_{k+1}$	Subexpression used in Theorem 4 ($k \geq 0$)
$\boldsymbol{\mathcal{S}}(p, q)$	Operator that circularly shifts a matrix by $p-1$ rows and $q-1$ columns
T	Total WTU cost incurred up to iteration $k \geq 0$
t_0	Compound input parameter of ACGM in extrapolated form
t_c^{LSSC}	LSSC mis-prediction correction cost in WTU
t_c^{MC}	MC mis-prediction correction cost in WTU
t_d^{LSSC}	LSSC mis-prediction detection cost in WTU
t_d^{MC}	MC mis-prediction detection cost in WTU
t_F	WTU cost of one call to $F(\boldsymbol{x})$
t_f	WTU cost of one call to $f(\boldsymbol{x})$
$T_{f,\Psi,L}(\boldsymbol{y})$	Proximal gradient operator of objective $f + \Psi$ with step size L at point $\boldsymbol{y}$
t_g	WTU cost of one call to $\nabla f(\boldsymbol{x})$
t_{k+1}	Vertex extrapolation term in ACGM ($k \geq 0$)
t_p	WTU cost of one call to $\mathrm{prox}_{\tau\Psi}(\boldsymbol{x})$
t_T	WTU cost of one call to $T_{f,\Psi,L}(\boldsymbol{y})$
t_Ψ	WTU cost of one call to $\Psi(\boldsymbol{x})$
U_k	Accuracy criterion upper bound / ISDUB estimate ($k \geq 1$)
u_{k+1}^*	Optimal value of $u_{k+1}(\boldsymbol{x})$ ($k \geq 0$)
$u_{k+1}(\boldsymbol{x})$	Local upper bound on the objective ($k \geq 0$)
$\boldsymbol{v}$	Vertex of parabola ψ
$\boldsymbol{v}_k$	Estimate function vertex ($k \geq 0$)
V_{k+1}	Subexpression used in Theorem 4 ($k \geq 0$)
$\boldsymbol{\mathcal{W}}$	Window operator parametrized by the top-left (i_1, j_1) and bottom-right coordinates (i_2, j_2) of the crop rectangle as well as the size of the input image (m_a, n_a)
W_k	Global lower bound incorporated in the estimate function ($k \geq 1$)
$w_{k+1}(\boldsymbol{x})$	Global lower bound on the objective used to update the estimate function ($k \geq 0$)
$\boldsymbol{\mathcal{W}}_{L,H}$	Shorthand for $\boldsymbol{\mathcal{W}}(m_L, m_H, n_L, n_H, m_N, n_N)$ where indices $L, H \in \{1, k, a, M, N\}$
$\boldsymbol{\mathcal{W}}_s(i_1, i_2)$	Shorthand for $\boldsymbol{\mathcal{W}}(i_1, i_2, 1, n_s, m_s, n_s)$ where index $s \in \{t, p\}$
$\boldsymbol{x}$	Vector of optimization variables / TRF to be recovered

$\boldsymbol{X}^*$	Set of optimal points
$\boldsymbol{x}^*$	Optimal point
$\boldsymbol{x}_{-1}$	Artificial iterate equal to $\boldsymbol{x}_0$
$\boldsymbol{x}_0$	Starting point of the algorithm
$\tilde{\boldsymbol{x}}_k$	Cached value of $\boldsymbol{A}\boldsymbol{x}_k$ ($k \geq 0$)
$\boldsymbol{x}_{k+1}$	Main iterate ($k \geq 0$)
$\boldsymbol{y}$	Observed radio-frequency image
$\tilde{\boldsymbol{y}}_k$	Cached value of $\boldsymbol{A}\boldsymbol{y}_k$ ($k \geq 0$)
$\boldsymbol{Y}_{k+1}$	Subexpression used in Theorem 4 ($k \geq 0$)
$\boldsymbol{y}_{k+1}$	Auxiliary point (control point) used in generating upper bound u_{k+1} ($k \geq 0$)
$\bar{\boldsymbol{y}}_{k+1}$	Vertex of the parabola component within the upper bound generated by $\boldsymbol{y}_{k+1}$
$\boldsymbol{\mathcal{Z}}$	Zero padding operator parametrized in the same way as $\boldsymbol{\mathcal{W}}$
$\boldsymbol{z}_0$	Artificial point equal to $\boldsymbol{x}_0$
$\boldsymbol{z}_{k+1}$	Result of applying a proximal gradient step at y_{k+1} ($k \geq 0$)
$\boldsymbol{\mathcal{Z}}_{L,H}$	Shorthand for $\boldsymbol{\mathcal{Z}}(m_L, m_H, n_L, n_H, m_N, n_N)$ where indices $L, H \in \{1, k, a, M, N\}$
$\boldsymbol{\mathcal{Z}}_s(i_1, i_2)$	Shorthand for $\boldsymbol{\mathcal{Z}}(i_1, i_2, 1, n_s, m_s, n_s)$ where index $s \in \{t, p\}$
α_k^{FGM}	Coefficient used in Nesterov's original FGM formulation
β_k	Auxiliary point extrapolation factor ($k \geq 0$)
γ	Curvature of parabola ψ
Γ_k	Augmented estimate sequence gap ($k \geq 0$)
γ_k	Estimate function curvature ($k \geq 0$)
Δ_k	Gap sequence term ($k \geq 0$)
$\bar{\Delta}_k$	Normalized Δ_k
$\theta(i, q)$	Kernel blending factor at row i next to the qth prototype
$\theta_{k,i}$	Coefficient of the gradient at iterate $\boldsymbol{x}_i$ used during iteration $k \geq 0$
λ_1	l_1-norm regularization factor
λ_2	Squared l_2-norm regularization factor
λ_k	Scalar term part of Nesterov's original estimate sequence definition ($k \geq 0$)
μ	Strong convexity of composite objective F
μ_f	Strong convexity parameter of smooth part f
$\mu_{f'}$	Strong convexity parameter of f'
μ_x	Lateral coordinate of the kernel center
μ_z	Axial coordinate of the kernel center
μ_Ψ	Strong convexity parameter of regularizer Ψ
ξ_{k+1}	A subgradient of Ψ at $\boldsymbol{x}_{k+1}$ related to the composite gradient
ξ'_{k+1}	A subgradient of Ψ' at $\boldsymbol{x}_{k+1}$ related to the composite gradient
$\rho_{\mu,\sigma}(x)$	Normalized Gaussian window with mean μ and standard deviation σ at x

σ_1	Minimal standard deviation
σ_2	Maximal standard deviation
$\sigma_{max}(\boldsymbol{A})$	Largest singular value of $\boldsymbol{A}$
$\sigma_x(i_h)$	Lateral standard deviation at row i_h
$\sigma_X(\boldsymbol{x})$	Indicator function of set X
σ_z	Axial standard deviation
τ	Proximal operator / proximal gradient step size
$\tau^{\mathrm{FISTA-CP}}$	Theoretically optimal step size of FISTA-CP
τ_{k+1}	Step size of ACGM ($k \geq 0$)
$\mathcal{T}_\tau(\boldsymbol{x})$	Shrinkage operator with step size τ
ϕ	Data fidelity term
ϕ_k	Normalized estimate function ($k \geq 0$)
Ψ'	Regularizer Ψ with all strong convexity removed
ψ^*	Optimal value of parabola ψ
ψ_k^*	Optimum value of ψ_k
$\psi_k'^*$	Optimum value of ψ_k'
$\psi_k(\boldsymbol{x})$	Estimate function ($k \geq 0$)
$\psi_k'(\boldsymbol{x})$	Augmented estimate function ($k \geq 0$)
$\Psi(\boldsymbol{x})$	Simple regularizer part of composite objective F
$\psi(\boldsymbol{x})$	Generic parabola
$\bar{\psi}_{\boldsymbol{x},\tau}(\boldsymbol{z})$	Parabola within the definition of the proximal operator, with vertex $\boldsymbol{x}$ and step size τ
ω_k	Coefficient of vertex $\boldsymbol{v}_k$ in the auxiliary point update
$\Omega^{\mathcal{M}}$	Line-search overhead of method $\mathcal{M}$
∞	Positive infinity
∇	Gradient
$\nabla_{\boldsymbol{x}}$	Gradient with respect to $\boldsymbol{x}$
$\hat{X}$	Current candidate quantity (scalar or vector) X
X_k	Quantity X at the beginning of iteration k
X_{k+1}	New value of quantity X generated during iteration k
$\lvert.\rvert$	Absolute value
$\lVert.\rVert_2$	Euclidean norm
$(.)_+$	Negative values are truncated to zero (maximum between argument and zero)
$(.)^T$	Matrix transpose / linear operator adjoint
$(.)^H$	Hermitian adjoint
$(.)^*$	Complex element-wise conjugate
$\oplus_c$	Circular sum applied to the set $\{1, ..., c\}$
$\ominus_c$	Circular difference applied to the set $\{1, ..., c\}$
$\circledast$	Discrete circular convolution
$*_1$	Discrete full convolution
$*_2$	Discrete valid convolution
$\otimes$	Kronecker product
$\odot$	Hadamard (element-wise) product

1. Introduction

1.1 Large-scale Optimization

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). Due to their size, these applications can be modeled as large-scale optimization problems for which simple operations such as the first-order derivative of objective function are computationally tractable but complex operations such as Hessian inversion are not [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.

Among large-scale applications, a broad range of problems, including the most common constrained smooth optimization problems, many inverse problems [5], and several classification and reconstruction problems in imaging [6] have a composite structure whereby the objective is a 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., [7–9]) and has a negligible cost compared to other operations. While many specialized methods have been introduced to tackle composite problems that have additional structure, such as sparsity (e.g., [10–13]), only small number of methods are applicable *to the entire problem class*.

These methods follow the black-box oracle model [14], which assumes that the exact structure of the objective function is not known to the optimization algorithm (outside the assumptions of the problem class) and algorithms can only obtain information on the problem by calling *oracle functions*. Apart from generality and theoretical simplicity, this model is 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.

1.2 Motivation

Nesterov has demonstrated that first-order methods can be accelerated, when he proposed his breakthrough Fast Gradient Method (FGM) [15]. FGM was constructed using the simple mathematical machinery of the *estimate sequence* [16]. 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 providing a guarantee of convergence.

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 [16]. However, FGM requires that the objective be continuously differentiable with Lipschitz gradient, the Lipschitz constant be known in advance, and the problem be unconstrained.

To address the demand for fast algorithms applicable to composite problems, which can have non-differentiable objectives and simple constraints, as well as to alleviate the need to know L_f in advance, Nesterov has introduced the Accelerated Multistep Gradient Scheme (AMGS) [17] that relies on *composite gradients* to overcome the limitations of FGM. This algorithm adjusts an estimate of L_f at every step (a process often called "line-search" in the literature [5, 18]) 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 computational complexity (see Subsection 4.4.2 for a detailed analysis).

The Fast Iterative Shrinkage-Thresholding Algorithm (FISTA) [5] decouples the advancement phase from the adjustment phase, stalling the former phase only during backtracks. However, FISTA has a fixed $\mathcal{O}(1/k^2)$ provable convergence rate even when the objective is strongly convex, and its line-search strategy cannot decrease the L_f estimate. Similar algorithms to FISTA have been collectively analyzed in [19], but none overcome

these drawbacks.

A strongly convex generalization of FISTA, which we designate by FISTA-Chambolle-Pock (FISTA-CP), was introduced in [6]. It has the same convergence guarantees as FGM in both the non-strongly and the strongly convex cases. The monograph [6] 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 [20] 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 [20] but weakens the theoretical convergence guarantees, making them poorer than those of AMGS (see also Subsection 4.4.2). The two restart heuristics proposed in [20] are altogether incompatible with the convergence analysis.

The second work [21] 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.

1.3 Objectives and Scope

The first objective of this work is to develop an *algorithm* that is applicable without modification to the entire class of composite problems. For many composite problems, a global Lipschitz constant may be difficult to compute or may be far larger than local values. Therefore, this new method should be able to dynamically estimate the local Lipschitz constant and use this estimate to increase the speed of convergence. In the worst case, the convergence rate should be no poorer than FGM.

Moreover, this method should also be able to produce a sequence of iterates with monotonically decreasing objective function values. Monotonicity prevents divergence in algorithms that employ proximal operators without a closed form expression or other kinds of inexact oracles [6, 22]. Even when dealing with exact oracles, monotonicity leads to a more stable and predictable convergence rate.

Existing methods applicable to composite problems are surprisingly

similar in form. However, apart from Nesterov's methods, the convergence analysis of each method seems independent of all others. A second objective is to provide a *unifying convergence analysis* for a large subset of the existing algorithms, preferably involving the estimate sequence or a notion derived from it.

Aside from difficulties in analysis, a major limiting factor of existing algorithms is the lack of a consistent design philosophy. The introduction of each method, with the notable exception of Nesterov's FGM, is not accompanied by a derivation. The absence of a derivation hinders the improvement and adaptation of an optimization method beyond its original scope. Therefore, the third objective of this thesis is to provide a simple and clear *design framework* for fast large-scale optimization algorithms.

1.4 Contributions

The contributions of this work span two previously unrelated fields.

1.4.1 Large-scale Convex Optimization Algorithms

1. 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 employed separately.

2. 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.

3. 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*.

4. 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 a composite objective structure, which has the convergence guarantees of FGM in both the non-strongly and strongly convex cases. ACGM is equipped with an explicit adaptive line-search procedure that is decoupled from the advancement phase at every iteration. Therefore, ACGM does not require a priori knowledge of the Lipschitz constant and

can converge when the Lipschitz property holds only locally.

5. We further showcase the flexibility and power of our framework by endowing ACGM with monotonicity alongside its adaptive line-search procedure.

6. ACGM is derived in a form related to the estimate sequence and can be brought to an equivalent form based on extrapolation that is more similar to FISTA and FISTA-CP.

7. With its variety of forms, ACGM encompasses FGM, FISTA, and FISTA-CP, along with their variants, while surpassing all these methods in terms of flexibility and usability.

8. We introduce the wall-clock time unit (WTU), a complexity measure that accounts for variations in per-iteration complexity of black-box optimization algorithms. WTU more accurately reflects the actual performance of such algorithms on practical applications.

9. When analyzed using the standard form of the WTU, ACGM has the best provable convergence rate among its class of algorithms both in the strongly and non-strongly convex cases.

10. We corroborate the theoretical findings with simulation results. Specifically, we show that our method surpasses the state-of-the-art when measured using both standard WTU and the most common form of generalized WTU. ACGM is particularly well suited for certain ultrasound image reconstruction problems, which can be effectively cast as strongly-convex large-scale composite problems.

1.4.2 Ultrasound Image Reconstruction

1. We propose two models based on spatially varying kernel convolution that accurately reflect the physics of ultrasound image formation.

2. Both models are linear and may be implemented as a matrix. However, the matrix forms do not scale because their complexity is proportional to the square of the number of pixels in the image. Therefore, we provide efficient matrix-free implementations of the model operators that entail the same computational cost as spatially invariant convolution.

3. The reconstruction problem is ill-posed and can only be addressed by employing first-order operations. The most computationally demanding

of these is the gradient of a data fidelity term. For both linear models, the data fidelity gradient expression includes calls to both the model operator and its adjoint. We provide matrix-free expressions for the adjoint operators of our models that are of equal complexity to the corresponding forward models. Our derivations are based on fundamental (and to our knowledge novel) theoretical results on discrete convolution.

4. By optimizing ACGM for this reconstruction problem, we are able to approximately half its per-iteration complexity.

5. We confirm using simulation results that reconstruction with ACGM and our models is tractable even for large images and produces results superior to those obtained using the spatially invariant model.

1.5 Author's Independent Contribution

This thesis summarizes the contents of five academic works (Publications I-V). These works comprise three publications in international journals with a review process (Publications II, III, and V) and two international conference publications (Publications I and IV). Two of these journal articles (Publications II and V) and both conference papers have undergone peer-review and have been published. One journal article (Publication III) has been submitted for publication and is currently under review. The author of this thesis is the main author of all above mentioned works and has undertaken the theoretical studies, including algorithm development, as well as the numerical simulations therein. The co-authors have supervised the main author, by helping with research planning, the writing as well as the revising of the thesis and the associated academic works.

1.6 Thesis Structure

This thesis contains seven chapters, which provide a unified view of the results described in the attached Publications I-V. Chapter 1 briefly describes the background and scope of this thesis, along with the author's contributions. In Chapter 2, we introduce the class of large-scale composite problems and use the theoretical worst-case performance bounds on this class to derive the estimate sequence, its augmentation, and the gap sequence. In Chapter 3, we develop from the fundamental structure of the composite problem class a design pattern for first-order algorithms. Using this pattern and the gap sequence, we derive the Accelerated Composite Gradient Method (ACGM). In Chapter 4, we provide a convergence analy-

sis for ACGM and discuss the theoretical relationship between our method and the competing methods. For a more realistic comparison, we introduce the wall-clock time unit (WTU), and use it to demonstrate the theoretical superiority of ACGM within its class. We support the theoretical results in Chapter 4 using an extensive simulation benchmark in Chapter 5. We discuss the reconstruction of ultrasound images in Chapter 6 and argue why ACGM is particularly well suited for this application. Chapter 7 concludes the thesis.

2. Augmenting the Estimate Sequence

When designing optimization algorithms, it is necessary to assume that not all information on a problem is available beforehand. Otherwise, the problem would be uniquely identifiable and the most accurate and efficient method that can be used to address it would consist of an algorithm that simply outputs the solution of that problem. The lack of complete information means that the available a priori information defines a collection of problems that share a common structure, which we call a problem *class*. It is in the context of such a problem class that we design numerical schemes and measure their performance [16].

2.1 The Large-scale Composite Problem Class

In this work, we consider the class of problems that have the following structure:

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} F(\boldsymbol{x}) \stackrel{\text{def}}{=} f(\boldsymbol{x}) + \Psi(\boldsymbol{x}), \tag{2.1}$$

where $\boldsymbol{x}$ is a vector of n optimization variables. The problem is *large-scale* [4] in the sense that first-order operations such as function gradients are computationally tractable to compute and manipulate in memory whereas second-order operation such as Hessians are intractable. 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 strong convexity parameter $\mu_f \geq 0$. The Lipschitz constant L_f may not be known to the optimization algorithm. The regularizer $\Psi : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ is a proper lower semicontinuous convex function with 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. Unlike f, Ψ need not be differentiable. However, its proximal map, given by

$$\mathbf{prox}_{\tau\Psi}(\boldsymbol{x}) \stackrel{\text{def}}{=} \operatorname*{arg\,min}_{\boldsymbol{z}\in\mathbb{R}^n} \left(\Psi(\boldsymbol{z}) + \frac{1}{2\tau}\|\boldsymbol{z}-\boldsymbol{x}\|_2^2 \right), \tag{2.2}$$

for all $\boldsymbol{x} \in \mathbb{R}^n$ and $\tau > 0$ can be computed with complexity $\mathcal{O}(n)$.

Apart from the above a priori information, the problem is treated by the algorithm as a *black box* [14]. Additional information on the problem can only be obtained by querying a number of *oracle functions*. The term oracle, alluding to the Oracle of Delphi in Ancient Greece, designates an abstract entity that has complete information on the problem but can only reveal a limited amount upon request, with each request bearing a cost.

The oracle functions pertaining to composite problems comprise $f(\boldsymbol{x})$, $\nabla f(\boldsymbol{x})$, $\Psi(\boldsymbol{x})$, and $\operatorname{prox}_{\tau\Psi}(\boldsymbol{x})$, with arguments $\boldsymbol{x} \in \mathbb{R}^n$ and $\tau > 0$.

Note that coordinate descent methods (CDMs) (e.g., [23, 24]) violate the aforementioned oracle model by additionally assuming that the objective is separable and therefore parts of $\nabla f(\boldsymbol{x})$ can be computed independently of all others. CDMs require a much larger number of iterations to converge compared to first-order methods and need to compensate by utilizing functional primitives of very low cost. Although CDMs show promise in a number large-scale applications with differentiable objectives [24], we exclude them from our analysis due to their reliance on additional problem structure.

2.2 Complexity Bounds

Throughout this work, we will assume that an optimization algorithm produces an iterate sequence $\{\boldsymbol{x}_{k+1}\}_{k\geq 0}$ of increasingly accurate (according to some criterion) estimates of an optimal point $\boldsymbol{x}^*$. By their very nature, iterations cannot be performed in parallel, even if computations within an iteration can be parallelized. The processing speed of computer systems has recently reached a saturation level, with virtually all advances in computer design revolving around parallelism [25]. Individual iterations may benefit from parallelization, but serial computational requirements, along with the synchronization and communication overhead, place a lower limit on the latency of an iteration (e.g., given by Amdahl's law [25]). Therefore, to keep the overall running time of the algorithm within practical limits, only a small number of iterations $K \ll n$ can be performed.

A natural means of defining the accuracy criterion is in the form of a simple upper bound U_k at every iteration k on the distance from the current iterate to the optimal set, namely

$$\inf_{\boldsymbol{x}^* \in \boldsymbol{X}^*} \|\boldsymbol{x}_k - \boldsymbol{x}^*\|_2^2 \leq U_k, \quad k \geq 0. \tag{2.3}$$

For the upper bounds to measure convergence, they must monotonically converge to zero. Monotonicity is required by our assumption on the increasing accuracy of the iterate sequence. Note that monotonicity applies only to the upper bounds, not to the actual distance between the iterates and the optimal set. Convergence to zero means that, for an arbitrarily

low accuracy $\epsilon > 0$, there must be an iteration k such that $U_i < \epsilon$ for all $i \geq k$. However, in almost all applications, several digits of accuracy are required for at most an order of thousands of iterations. Hence, for the upper bounds to be of practical importance, they should belong to a complexity class $\mathcal{O}(1/k^p)$, $p \geq 1$ (sub-linear), $\mathcal{O}(e^{-k})$ (linear), or better.

2.3 Convergence Guarantees

Nesterov has derived in [16] the following result applicable to the class of non-strongly convex differentiable problems with Lipschitz gradient $\mathcal{F}_{L_f}^{\infty,1}(\mathbb{R}^n)$. This class consists of problems that take the form of

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} F(\boldsymbol{x}) = f(\boldsymbol{x}), \tag{2.4}$$

where n, $\boldsymbol{x}$, and f have the same properties as in the composite problem class, with the exception that μ_f is always 0.

Theorem 1. *For any scheme that produces at every iteration $k \geq 0$ the sequence of parameters $\{\theta_{k,i}\}_{0\leq i\leq k}$ and a new iterate, given by*

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_0 + \sum_{i=0}^{k} \theta_{k,i} \nabla f(\boldsymbol{x}_k),$$

there exists a function $f(\boldsymbol{x})$ such that

$$\|\boldsymbol{x}_k - \boldsymbol{x}^*\|_2^2 \geq \frac{1}{8}\|\boldsymbol{x}_0 - \boldsymbol{x}^*\|_2^2,$$

for all $\boldsymbol{x}^* \in \boldsymbol{X}^*$ as long as $k \leq \frac{n-1}{2}$.

Proof. The reasoning is based on information theoretical arguments. Nesterov provides in [16] an ill-conditioned quadratic objective f where, for any problem, a coordinate system is imposed such that the starting point $\boldsymbol{x}_0$ is the origin. Each gradient adds exactly one to the dimensionality of the linear span of all previous iterates. The lower bound gives the distance to that subspace. See [16] for a detailed analysis. □

The bound on the iterate convergence in Theorem 1 can be shown to hold for more sophisticated full gradient schemes due to the limited amount of information revealed by gradient calls [16].

Every problem in $\mathcal{F}_{L_f}^{\infty,1}(\mathbb{R}^n)$ is a composite problem that includes two additional assumptions: $\Psi(\boldsymbol{x}) = 0$ for all $x \in \mathbb{R}^n$ and $\mu = \mu_f = 0$. Therefore, $\mathcal{F}_{L_f}^{\infty,1}(\mathbb{R}^n)$ is a subset of the composite problem class and the lower bounds in Theorem 1 apply to composite problems as well.

However, the result in Theorem 1 effectively invalidates our previous accuracy criterion in (2.3). Note that the eventual convergence of iterates on composite problems has been proven in [26] for a variant of FISTA

but such a result is not of practical importance. The number of iterations required even for several digits of accuracy is prohibitive (as argued by Theorem 1), regardless of the computing power of the system on which the algorithm runs. Thus, we need to devise a different accuracy criterion for our problem class. The following result, obtained in [16], leads to an alternative.

Theorem 2. *Under the same assumptions as in Theorem 1, we have that*

$$F(\boldsymbol{x}_k) - F(\boldsymbol{x}^*) \geq \max\left\{\frac{3L_f}{32(k+1)^2}, \frac{\mu}{2}\left(\frac{1-\sqrt{q}}{1+\sqrt{q}}\right)^{2k}\right\}\|\boldsymbol{x}_0 - \boldsymbol{x}^*\|_2^2,$$

for all $\boldsymbol{x}^* \in \boldsymbol{X}^*$. Here, q is the inverse condition number given by $q = \frac{\mu}{L_f}$.

Proof. For $\mu = 0$, the ill-conditioned quadratic objective f in the proof of Theorem 1 produces the bound on the objective value decrease. When $\mu > 0$, $f(\boldsymbol{x}) + \frac{\mu}{2}\|\boldsymbol{x}\|_2^2$ is used in the same way. See [16] for a complete exposition. □

Theorem 2 compels us to express the convergence rate of first-order schemes on composite problems in a similar manner, i.e. using the image space distance (ISD), which is the distance between the composite objective values at the iterates and the optimal value. We therefore define a convergence guarantee (provable convergence rate) as the *decrease rate* of a theoretical upper bound on this ISD.

The upper bounds provided by Nesterov for FGM in [16] contain, apart from the domain space term in Theorem 2, the initial ISD. Given that our current aim is to provide a generic framework that applies to FGM and its variants, we include the weighted initial ISD in our upper bounds. Thus, we formulate the image space distance upper bound (ISDUB) as

$$A_k(F(\boldsymbol{x}_k) - F(\boldsymbol{x}^*)) \leq A_0(F(\boldsymbol{x}_0) - F(\boldsymbol{x}^*)) + \frac{\gamma_0}{2}\|\boldsymbol{x}_0 - \boldsymbol{x}^*\|_2^2, \tag{2.5}$$

for all $\boldsymbol{x}^* \in X^*$ and $k \geq 0$. Here, the convergence guarantees are given by the sequence $\{A_k\}_{k\geq 0}$. Convergence guarantees are only meaningful if positive. However, in algorithms which allow $\boldsymbol{x}_0$ to be infeasible, A_0 must be zero. The assumption of increasingly accurate iterates implies that the convergence guarantees are monotonically increasing. Therefore, the convergence guarantees obey $A_0 \geq 0$, $A_k > 0$ for all $k \geq 1$, and $A_{k+1} > A_k$ for all $k \geq 0$.

The right-hand side of (2.5) is a weighted sum between the initial ISD and the corresponding domain space term, with weights given by A_0 and γ_0, respectively. The possibility of A_0 being zero is what motivates us to write the ISDUB expression (2.5) in a form in which the objective value at the current iterate is weighted by the convergence guarantee, which differs from the bounds formulated in Theorem 2. We impose no restrictions on the weights in (2.5), apart from $A_0 \geq 0$ and $\gamma_0 > 0$. The positivity of γ_0 is

required by Theorem 2 and also holds significance in our interpretation of the estimate sequence, along with its augmented variant, as we shall demonstrate in the sequel.

2.4 The Estimate Sequence

2.4.1 A Substitute Convergence Guarantee

The ISDUB expression in (2.5) can be rearranged to take the form

$$A_k F(\boldsymbol{x}_k) \leq H_k, \quad k \geq 0, \tag{2.6}$$

where

$$H_k \stackrel{\text{def}}{=} (A_k - A_0)F(\boldsymbol{x}^*) + A_0 F(\boldsymbol{x}_0) + \frac{\gamma_0}{2}\|\boldsymbol{x}_0 - \boldsymbol{x}^*\|_2^2, \quad k \geq 0 \tag{2.7}$$

is the highest upper bound that can be placed on weighted objective values $A_k F(\boldsymbol{x}_k)$ to satisfy (2.5). The value of H_k depends on the optimal point $\boldsymbol{x}^*$, which is an unknown quantity. Note that for non-strongly convex objectives, $\boldsymbol{x}^*$ may not be unique. Without loss of generality, we will fix $\boldsymbol{x}^*$ to be an arbitrary element of $\boldsymbol{X}^*$ throughout the remainder of this work.

The estimate sequence (ES) provides a computable, albeit more stringent, replacement for H_k. It is obtained as follows. The convexity of the objective implies the existence of a sequence $\{W_k\}_{k\geq 1}$ of convex global lower bounds on F, namely

$$F(\boldsymbol{x}) \geq W_k(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^n, \quad k \geq 1. \tag{2.8}$$

By substituting the optimal value terms $F(\boldsymbol{x}^*)$ in (2.6) with $W_k(\boldsymbol{x}^*)$, we obtain $\mathcal{H}_k$, a lower bound on H_k, given by

$$\mathcal{H}_k \stackrel{\text{def}}{=} (A_k - A_0)W_k(\boldsymbol{x}^*) + A_0 F(\boldsymbol{x}_0) + \frac{\gamma_0}{2}\|\boldsymbol{x}^* - \boldsymbol{x}_0\|_2^2, \quad k \geq 0. \tag{2.9}$$

Note that H_0 does not depend on $F(\boldsymbol{x}^*)$ and $\mathcal{H}_0$ is the same as H_0. Therefore, it is not necessary to define W_0, which explains why we have $k \geq 1$ in (2.8). For $k \geq 1$, $\mathcal{H}_k$ is difficult to enforce as an *upper bound* on $A_k F(\boldsymbol{x}_k)$, partly because of its dependence on $\boldsymbol{x}^*$. However, $\mathcal{H}_k$ can be viewed as the value of an *estimate function* at $\boldsymbol{x}^*$. The estimate functions $\psi_k(\boldsymbol{x})$, $k \geq 0$ can be thus be modeled as functional extensions of $\mathcal{H}_k$, namely

$$\psi_k(\boldsymbol{x}) \stackrel{\text{def}}{=} (A_k - A_0)W_k(\boldsymbol{x}) + A_0 F(\boldsymbol{x}_0) + \frac{\gamma_0}{2}\|\boldsymbol{x} - \boldsymbol{x}_0\|_2^2, \quad \boldsymbol{x} \in \mathbb{R}^n, \quad k \geq 0. \tag{2.10}$$

We define the *estimate sequence* as the collection of estimate functions $\{\psi_k(\boldsymbol{x})\}_{k\geq 0}$.

The first estimate function ψ_0 is given by

$$\psi_0(\boldsymbol{x}) = A_0 F(\boldsymbol{x}_0) + \frac{\gamma_0}{2}\|\boldsymbol{x} - \boldsymbol{x}_0\|_2^2, \quad \boldsymbol{x} \in \mathbb{R}^n. \tag{2.11}$$

Substituting (2.11) in definition (2.10) gives a simple and general form for estimate functions, stated as

$$\psi_k(\boldsymbol{x}) = (A_k - A_0)W_k(\boldsymbol{x}) + \psi_0(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^n, \quad k \geq 0. \tag{2.12}$$

The estimate function optimum value, given by

$$\psi_k^* \stackrel{\text{def}}{=} \min_{\boldsymbol{x} \in \mathbb{R}^n} \psi_k(\boldsymbol{x}), \quad k \geq 0, \tag{2.13}$$

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

$$\psi_k^* = \min_{\boldsymbol{x} \in \mathbb{R}^n} \psi_k(\boldsymbol{x}) \leq \psi_k(\boldsymbol{x}^*) = \mathcal{H}_k, \quad k \geq 0. \tag{2.14}$$

As such, ψ_k^* provides the sought after computable replacement of H_k. Thus, it suffices to maintain the estimate sequence property (ESP), given by

$$A_k F(\boldsymbol{x}_k) \leq \psi_k^*, \quad k \geq 0, \tag{2.15}$$

to satisfy the ISDUB expression in (2.5). The proof follows from the above definitions as

$$A_k F(\boldsymbol{x}_k) \overset{(2.15)}{\leq} \psi_k^* \overset{(2.14)}{\leq} \psi_k(\boldsymbol{x}^*) = \mathcal{H}_k \overset{(2.8)}{\leq} H_k, \quad k \geq 0. \tag{2.16}$$

2.4.2 Nesterov's Estimate Sequences

As we have seen in Subsection 2.4.1, the construction of the estimate sequence follows analytically from the ISDUB expression in (2.5). However, at this stage, the relationship between the estimate functions and the objective is not clear. By contrast, Nesterov has derived the estimate sequence starting from the properties of finite objectives and has extended the definition to composite objectives in a similar form to (2.10).

Nesterov's original estimate sequence
The estimate sequence defined in [16] for $\mathcal{F}_{L_f}^{\infty,1}(\mathbb{R}^n)$ is actually a pair of sequences, $\{\phi_k(\boldsymbol{x})\}_{k\geq 0}$ and $\{\lambda_k\}_{k\geq 0}$, that satisfy the following: $\lambda_k \geq 0$ for $k \geq 0$, $\lim_{k\to\infty} \lambda_k = 0$, and

$$\phi_k(\boldsymbol{x}) \leq (1 - \lambda_k)F(\boldsymbol{x}) + \lambda_k \phi_0(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^n. \tag{2.17}$$

According to this definition, estimate functions $\phi_k(\boldsymbol{x})$ are arbitrarily accurate *approximate global lower bounds* on the objective F. The estimate sequence property is defined as

$$F(\boldsymbol{x}_k) \leq \phi_k^* \stackrel{\text{def}}{=} \min_{\boldsymbol{x} \in \mathbb{R}^n} \phi_k(\boldsymbol{x}), \quad k \geq 0. \tag{2.18}$$

This property states that the optimal values of the estimate functions, themselves global lower bounds on the estimate functions, are also *local upper bounds* on the objective at the iterates.

However, when designing optimization schemes, the assumptions on sequence $\{\lambda_k\}_{k\geq 0}$ need to be stricter.

Lemma 1. *To produce a valid optimization algorithm, we must have in Nesterov's definition of the estimate sequence* (2.17) *that* $\lambda_0 = 1$ *as well as* $\lambda_k > 0$ *and* $\lambda_k < \lambda_{k+1}$ *for all* $k \geq 0$.

Proof. For $k = 0$, (2.17) reduces to

$$(1 - \lambda_0)\phi_0(\boldsymbol{x}) \leq (1 - \lambda_0)F(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^n. \tag{2.19}$$

We assume that in (2.19) we have $\lambda_0 \neq 1$. Then, by dividing both sides of (2.19) by the non-zero $(1 - \lambda_0)$, we have that the initial estimate function $\phi_0(\boldsymbol{x})$ is a global lower bound on F. The estimate sequence property (2.18) gives

$$F(\boldsymbol{x}_0) \leq \phi_0^* = \min_{\boldsymbol{x} \in \mathbb{R}^n} \phi_0(\boldsymbol{x}) \leq \min_{\boldsymbol{x} \in \mathbb{R}^n} F(\boldsymbol{x}) = F(\boldsymbol{x}^*) \leq F(\boldsymbol{x}_0), \tag{2.20}$$

meaning that $\boldsymbol{x}_0 \in \boldsymbol{X}^*$, which precludes the need for a numerical scheme.

Combining (2.17) and (2.18), we obtain (see also Lemma 2.2.1 in [16]) that

$$F(\boldsymbol{x}_k) - F(\boldsymbol{x}^*) \leq \lambda_k(\phi_0(\boldsymbol{x}^*) - F(\boldsymbol{x}^*)). \tag{2.21}$$

If for a certain $\tilde{k} \geq 0$ we have $\lambda_{\tilde{k}} = 0$ then (2.21) implies that $F(\boldsymbol{x}_{\tilde{k}}) = F(\boldsymbol{x}^*)$, meaning that $\boldsymbol{x}_{\tilde{k}} \in \boldsymbol{X}^*$. Again, this contradicts our fundamental assumption of imperfect information on the problem class.

Moreover, (2.21) states that the sequence $\{\lambda_k\}_{k \geq 0}$ determines the convergence guarantees of the optimization scheme. For such a method to make meaningful progress at every iteration, it is necessary to have $\lambda_k < \lambda_{k+1}$ for all $k \geq 0$. □

Canonical form

Lemma 1 allows us to perform the following substitution:

$$\lambda_k \overset{\text{def}}{=} \frac{A_0}{A_k}, \quad \phi_k(\boldsymbol{x}) \overset{\text{def}}{=} \frac{1}{A_k}\psi_k(\boldsymbol{x}), \quad k \geq 0, \quad \boldsymbol{x} \in \mathbb{R}^n. \tag{2.22}$$

To avoid confusion between the two forms of the estimate sequence, we denote the terms $\phi_k(\boldsymbol{x})$ from now on as *normalized* estimate functions and $\psi_k(\boldsymbol{x})$ as estimate functions.

The estimate sequence definition becomes

$$\psi_k(\boldsymbol{x}) \leq (A_k - A_0)F(\boldsymbol{x}) + \psi_0(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^n, \quad k \geq 0, \tag{2.23}$$

with the convergence guarantees satisfying $A_k > 0$ and $A_{k+1} > A_k$ for all $k \geq 0$ with $\lim_{k \to \infty} A_k = \infty$. Therefore, at every iteration $k \geq 1$, there exists $W_k(\boldsymbol{x})$, a global lower bound on F, such that

$$\psi_k(\boldsymbol{x}) = (A_k - A_0)W_k(\boldsymbol{x}) + \psi_0(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^n, \quad k \geq 1. \tag{2.24}$$

The existence and computability of ϕ_k^* implies that the lower bounds $W_k(\boldsymbol{x})$ are convex. Note that Nesterov's estimate function expression in canonical

form (2.24) matches the one in (2.12). Moreover, the estimate sequence property in [16], given by

$$F(\boldsymbol{x}_k) \leq \phi_k^*, \quad k \geq 0 \tag{2.25}$$

is made equivalent to (2.15) by scaling with A_k, which are positive in this context for all $k \geq 0$.

Nesterov's original estimate sequence only differs from the canonical form in (2.12) by not accommodating infeasible start due to the implicit assumption $A_0 > 0$. It also does not place any restriction of the initial estimate function apart from the estimate sequence property (2.15). However, when designing FGM for $\mathcal{F}_{L_f}^{\infty,1}(\mathbb{R}^n)$ in [16], $\psi_0(\boldsymbol{x})$ is also set according to (2.11).

Nesterov's newer variant

Nesterov has addressed the infeasible start issue in [17] with an updated estimate sequence definition, given by

$$\psi_k(\boldsymbol{x}) = l_k(\boldsymbol{x}) + (A_k - A_0)\Psi(\boldsymbol{x}) + \psi_0(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^n, \quad k \geq 0, \tag{2.26}$$

where $l_k(\boldsymbol{x})$ is a linear function. In [17], $l_0(\boldsymbol{x})$ is set to 0 for all $\boldsymbol{x} \in \mathbb{R}^n$. At every iteration, the estimate sequence is updated as

$$\psi_{k+1}(\boldsymbol{x}) = \psi_k(\boldsymbol{x}) + a_{k+1}\tilde{l}_{k+1}(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^n, \quad k \geq 0, \tag{2.27}$$

where $\tilde{l}_{k+1}$ is a lower linear model of the function at $\boldsymbol{x}_{k+1}$, given by

$$\tilde{l}_{k+1}(\boldsymbol{x}) = f(\boldsymbol{x}_{k+1}) + \langle \nabla f(\boldsymbol{x}_{k+1}), \boldsymbol{x} - \boldsymbol{x}_{k+1} \rangle, \quad \boldsymbol{x} \in \mathbb{R}^n, \quad k \geq 0, \tag{2.28}$$

and weight a_{k+1} increments the convergence guarantees as

$$A_{k+1} = A_k + a_{k+1}, \quad k \geq 0. \tag{2.29}$$

It follows by induction that in [17] every linear function l_k for $k \geq 1$ is given by

$$l_k(\boldsymbol{x}) = \sum_{i=1}^{k} a_k \tilde{l}_i(\boldsymbol{x}) = (A_k - A_0) w_k(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^n, \tag{2.30}$$

where

$$w_k(\boldsymbol{x}) = \frac{\sum_{i=1}^{k} a_i \tilde{l}_i(\boldsymbol{x})}{\sum_{i=1}^{k} a_i}, \quad \boldsymbol{x} \in \mathbb{R}^n, \quad k \geq 1, \tag{2.31}$$

is a weighted average of lower linear models, each a global lower bound on objective F. Therefore w_k is itself a global lower bound on f. We set W_k for each $k \geq 1$ as

$$W_k(\boldsymbol{x}) = w_k(\boldsymbol{x}) + \Psi(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^n, \tag{2.32}$$

and obtain a global lower bound on F. From (2.32), we again obtain the canonical form in (2.12). The estimate sequence property is the same as in (2.15) and the initial estimate function is also given by (2.11), although the results derived in [17] apply also to non-standard Euclidean norms, which is beyond the scope of our work. We leave the generalization of our framework to such norms as an open topic for future research.

In the context of [17], A_0 is always zero, which is more restrictive than in our model and incompatible with Nesterov's original estimate sequence definition. Moreover, the newer variant no longer takes into account strong convexity. The latter restriction is imposed because many other problem classes (e.g., [27, 28]) do not involve strong convexity and the form in (2.26) generalizes easily to address them. For that matter, (2.26) was actually first introduced for the class of differentiable objectives ($\Psi = 0$) with Lipschitz continuous Hessians [28] under the assumption $\psi_0 = A_0 F(\boldsymbol{x}_0) + 2L_3\|\boldsymbol{x} - \boldsymbol{x}_0\|_2^3$ with $A_0 = 1$, where L_3 is the Hessian Lipschitz constant. Nonetheless, the fact that even a partially restricted variant of the estimate sequence can be applied to such a wide array of problem classes demonstrates the versatility and fundamental nature of the estimate sequence.

2.5 The Augmented Estimate Sequence

Recall that the estimate sequence property in (2.15) produces a gap between ψ_k^* and H_k, shown in (2.16). This allows us to introduce the more relaxed *augmented estimate sequence* (AES) $\{\psi'_k(\boldsymbol{x})\}_{k\geq 0}$ defined, using the notation and conventions from Subsection 2.4.1, as

$$\psi'_k(\boldsymbol{x}) \stackrel{\text{def}}{=} \psi_k(\boldsymbol{x}) + H_k - \mathcal{H}_k, \quad k \geq 0. \tag{2.33}$$

We expand definition (2.33) as

$$\psi'_k(\boldsymbol{x}) = \psi_k(\boldsymbol{x}) + (A_k - A_0)(F(\boldsymbol{x}^*) - W_k(\boldsymbol{x}^*)), \quad k \geq 0. \tag{2.34}$$

Augmentation therefore consists only of adding a non-negative constant (due to the lower bound property of W_k) to the estimate function, thus preserving its shape.

The augmented estimate sequence property (AESP) is given by

$$A_k F(\boldsymbol{x}_k) \leq \psi'^*_k, \quad k \geq 0. \tag{2.35}$$

This property can be used to derive the provable convergence rate because, along with definitions (2.9), (2.10), and (2.33), it implies that

$$A_k F(\boldsymbol{x}_k) \leq \psi'^*_k = \psi_k^* + H_k - \mathcal{H}_k = H_k + (\psi_k^* - \psi_k(\boldsymbol{x}^*)) \leq H_k, \quad k \geq 0. \tag{2.36}$$

The augmented estimate sequence property (2.35) can alternatively be written as

$$A_k F(\boldsymbol{x}_k) \leq \psi'_k(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^n, \quad k \geq 0. \tag{2.37}$$

Substituting (2.12) and (2.33) in (2.37) gives

$$A_k F(\boldsymbol{x}_k) \leq (A_k - A_0)(W_k(\boldsymbol{x}) - W_k(\boldsymbol{x}^*) + F(\boldsymbol{x}^*)) + \psi_0(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^n, \quad k \geq 0. \tag{2.38}$$

By expanding the initial estimate function as in (2.11) and rearranging terms we obtain

$$\begin{aligned} A_k(F(\boldsymbol{x}_k) - F(\boldsymbol{x}^*)) &\leq (A_k - A_0)(W_k(\boldsymbol{x}) - W_k(\boldsymbol{x}^*)) \\ +A_0(F(\boldsymbol{x}_0) - F(\boldsymbol{x}^*)) &+ \frac{\gamma_0}{2}\|\boldsymbol{x} - \boldsymbol{x}_0\|_2^2, \quad \boldsymbol{x} \in \mathbb{R}^n, \quad k \geq 0. \end{aligned} \tag{2.39}$$

The benefits of augmentation are now clearer. For instance, by setting $\boldsymbol{x} = \boldsymbol{x}^*$ in (2.39) we obtain the ISDUB property (2.5). This confirms that the estimate sequence optimum is a valid upper bound, as already demonstrated in (2.36). Moreover, the form in (2.39) is more robust than (2.15). *Independently* adding arbitrary constant terms to W_k for every $k \geq 1$ does not alter (2.39). This characteristic is inherited by the gap sequence, which we introduce in the sequel.

2.6 Parabolae

We define a parabola as a quadratic function $\psi : \mathbb{R}^n \to \mathbb{R}$ whose Hessian is a positive multiple of the identity matrix, namely

$$\psi(\boldsymbol{x}) \stackrel{\text{def}}{=} \psi^* + \frac{\gamma}{2}\|\boldsymbol{x} - \boldsymbol{v}\|_2^2, \quad \boldsymbol{x} \in \mathbb{R}^n, \tag{2.40}$$

where $\gamma > 0$ gives the curvature, $\boldsymbol{v} \in \mathbb{R}^n$ is the vertex, and $\psi^* \in \mathbb{R}$ is the optimal value. We denote the set of all parabolae as $\mathcal{P}$.

In this work, we define a hyperplane as a linear function $h : \mathbb{R}^n \to \mathbb{R}$ and the set of hyperplanes as $\mathcal{H}$. A generalized parabola is a function whose Hessian is a non-negative multiple of the identity matrix. Therefore, a generalized parabola is either a parabola or a hyperplane. The generalized parabola set is given by $\mathcal{G} \stackrel{\text{def}}{=} \mathcal{P} \cup \mathcal{H}$.

Parabolae constitute an important building block in our analysis because the Lipschitz gradient and the strong convexity properties can be defined in terms of parabolic upper bounds and generalized parabolic lower bounds. To simplify notation, we define two abbreviated expressions, hyperplane $P_{f,\boldsymbol{y}}(\boldsymbol{x})$ and generalized parabola $Q_{f,\gamma,\boldsymbol{y}}(\boldsymbol{x})$, as

$$P_{f,\boldsymbol{y}}(\boldsymbol{x}) \stackrel{\text{def}}{=} f(\boldsymbol{y}) + \langle \nabla f(\boldsymbol{y}), \boldsymbol{x} - \boldsymbol{y} \rangle, \tag{2.41}$$

$$Q_{f,\gamma,\boldsymbol{y}}(\boldsymbol{x}) \stackrel{\text{def}}{=} P_{f,\boldsymbol{y}}(\boldsymbol{x}) + \frac{\gamma}{2}\|\boldsymbol{x} - \boldsymbol{y}\|_2^2, \tag{2.42}$$

for any $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ and $\gamma > 0$. Using this notation, we illustrate the upper and lower bounds for f resulting from the definition of Lipschitz gradient and strong convexity as

$$Q_{f,\mu_f,\boldsymbol{y}}(\boldsymbol{x}) \leq f(\boldsymbol{x}) \leq Q_{f,L_f,\boldsymbol{y}}(\boldsymbol{x}), \quad \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n. \tag{2.43}$$

Moreover, parabolae can be combined to create new bounds. The simplest way of constructing new lower and upper bounds from simple bounds such as the ones in (2.43) is by weighted averaging. The following result shows that these new bounds retain their basic properties and, as we shall see, can determine the structure of the estimate functions.

Lemma 2. *Let ψ_1, ψ_2 be generalized parabolae with curvatures γ_1 and γ_2, respectively. Then, for any $\alpha_1, \alpha_2 \geq 0$, $\alpha_1\psi_1 + \alpha_2\psi_2$ is a generalized parabola with the curvature given by $\alpha_1\gamma_1 + \alpha_2\gamma_2$.*

Proof. By definition

$$\psi_1, \psi_2 \in \mathcal{G} \quad \Leftrightarrow \quad \mathbf{H}\psi_1(\boldsymbol{x}) = \gamma_1 \boldsymbol{I}_n, \quad \mathbf{H}\psi_2(\boldsymbol{x}) = \gamma_2 \boldsymbol{I}_n, \tag{2.44}$$

with $\gamma_1 \geq 0$ and $\gamma_2 \geq 0$. The Hessian is a linear operator, therefore

$$\begin{aligned} \mathbf{H}(\alpha_1\psi_1 + \alpha_2\psi_2) &= \alpha_1\mathbf{H}\psi_1 + \alpha_2\mathbf{H}\psi_2 \overset{(2.44)}{=} \alpha_1(\gamma_1\boldsymbol{I}_n) + \alpha_2(\gamma_2\boldsymbol{I}_n) \\ &= (\alpha_1\gamma_1 + \alpha_2\gamma_2)\boldsymbol{I}_n. \end{aligned} \tag{2.45}$$

Because $\alpha_1\gamma_1 + \alpha_2\gamma_2 \geq 0$ we have that $\alpha_1\psi_1 + \alpha_2\psi_2 \in \mathcal{G}$. See [16] for more details. □

2.6.1 Parabolic Estimate Functions

When the objective is non-strongly convex, the only available simple global convex lower bounds are hyperplanes. If we further assume that new bounds are obtained only by weighted averaging simple bounds, W_k become linear in (2.10) for all $k \geq 0$. The initial estimate function, given by (2.11), is a generalized parabola. Lemma 2, combined with the general form of the estimate function in (2.10), implies that when W_k are linear, every estimate function $\psi_k(\boldsymbol{x})$ is a generalized parabola with the curvature given by γ_0. The existence of ψ_k^* in this case is conditioned by $\gamma_0 > 0$.

Conversely, once we assume that $\gamma_0 > 0$ and that all lower bounds W_k are generalized parabolae, the estimate functions become parabolae. Since augmentation consists of adding a constant factor, every estimate function has the same curvature and vertex as its augmented counterpart. We write the estimate functions and the augmented estimate functions in parabolic form (2.40) with vertex $\boldsymbol{v}_k$ and curvature γ_k for all $k \geq 0$ as

$$\psi_k(\boldsymbol{x}) = \psi_k^* + \frac{\gamma_k}{2}\|\boldsymbol{x} - \boldsymbol{v_k}\|_2^2, \tag{2.46}$$

$$\psi_k'(\boldsymbol{x}) = \psi_k'^* + \frac{\gamma_k}{2}\|\boldsymbol{x} - \boldsymbol{v_k}\|_2^2. \tag{2.47}$$

From (2.34), we have that

$$\psi_k'^* = \psi_k^* + (A_k - A_0)(F(\boldsymbol{x}^*) - W_k(\boldsymbol{x}^*)). \tag{2.48}$$

The initial estimate function (2.11) does not contain a lower bound term, hence $\boldsymbol{v}_0 = \boldsymbol{x}_0$.

2.6.2 Composite Parabolae

For a given function Ψ, we also define the family of composite parabolae $\mathcal{P}_\Psi = \{\psi + \Psi \mid \psi \in \mathcal{P}\}$. Composite parabolae are closely connected to the proximal operator oracle function. Specifically, for any $\boldsymbol{x} \in \mathbb{R}^n$ and step size $\tau > 0$, let

$$\bar{\psi}_{\boldsymbol{x},\tau}(\boldsymbol{z}) \stackrel{\text{def}}{=} \frac{1}{2\tau}\|\boldsymbol{z} - \boldsymbol{x}\|_2^2, \quad \boldsymbol{z} \in \mathbb{R}^n. \tag{2.49}$$

Function $\bar{\psi}_{\boldsymbol{x},\tau}$ is obviously a parabola. The proximal operator $\operatorname{prox}_{\tau\Psi}$ therefore returns the unique optimum point of the composite parabola $\bar{\psi}_{\boldsymbol{x},\tau} + \Psi$.

Furthermore, the proximal operator can compute the optimum of *any* composite parabola, since by substituting $\gamma \stackrel{\text{def}}{=} 1/\tau$ and $\boldsymbol{v} \stackrel{\text{def}}{=} \boldsymbol{x}$, the definition of $\bar{\psi}_{\boldsymbol{x},\tau}$ is equivalent to that of a parabola.

2.7 The Gap Sequence

A sufficient condition for the preservation of the augmented estimate sequence property (2.35) across iterations is that the augmented estimate sequence gap, defined as

$$\Gamma_k \stackrel{\text{def}}{=} A_k F(\boldsymbol{x}_k) - \psi_k'^*, \quad k \geq 0, \tag{2.50}$$

is non-increasing.

When the lower bounds W_k are generalized parabolae, we have that

$$\psi(\boldsymbol{x}^*) \stackrel{(2.46)}{=} \psi_k^* + \frac{\gamma_k}{2}\|\boldsymbol{x}^* - \boldsymbol{x}_0\|_2^2, \tag{2.51}$$

$$\psi(\boldsymbol{x}^*) \stackrel{(2.34)}{=} (A_k - A_0)W_k(\boldsymbol{x}^*) + A_0 F(\boldsymbol{x}_0) + \frac{\gamma_0}{2}\|\boldsymbol{x}^* - \boldsymbol{v}_k\|_2^2, \tag{2.52}$$

for all $k \geq 0$. Combining the two forms of the estimate function gives

$$(A_k - A_0)W_k(\boldsymbol{x}^*) - \psi_k^* = \frac{\gamma_k}{2}\|\boldsymbol{x}^* - \boldsymbol{v}_k\|_2^2 - \frac{\gamma_0}{2}\|\boldsymbol{x}^* - \boldsymbol{x}_0\|_2^2 - A_0 F(\boldsymbol{x}_0), \tag{2.53}$$

for all $k \geq 0$. Therefore, the augmented estimate sequence gap can be written as

$$\begin{aligned}\Gamma_k &\stackrel{(2.48)}{=} A_k(F(\boldsymbol{x}_k) - F(\boldsymbol{x}^*)) + (A_k - A_0)W_k(\boldsymbol{x}^*) + A_0 F(\boldsymbol{x}^*) - \psi_k^* \\ &\stackrel{(2.53)}{=} A_k(F(\boldsymbol{x}_k) - F(\boldsymbol{x}^*)) + \frac{\gamma_k}{2}\|\boldsymbol{v_k} - \boldsymbol{x}^*\|_2^2 \\ &\quad - A_0(F(\boldsymbol{x}_0) - F(\boldsymbol{x}^*)) - \frac{\gamma_0}{2}\|\boldsymbol{x_0} - \boldsymbol{x}^*\|_2^2, \quad k \geq 0.\end{aligned} \tag{2.54}$$

We introduce the *gap sequence* $\{\Delta_k\}_{k\geq 0}$ in the form of

$$\Delta_k \stackrel{\text{def}}{=} A_k(F(\boldsymbol{x}_k) - F(\boldsymbol{x}^*)) + \frac{\gamma_k}{2}\|\boldsymbol{v}_k - \boldsymbol{x}^*\|_2^2, \quad k \geq 0. \tag{2.55}$$

The augmented estimate sequence gaps can be expressed more succinctly as

$$\Gamma_k = \Delta_k - \Delta_0, \quad k \geq 0. \tag{2.56}$$

Hence, the variation of the two sequences is identical, with the only difference being that the augmented estimate sequence gap is constrained to be zero initially. The sufficient condition becomes

$$\Delta_{k+1} \leq \Delta_k, \quad k \geq 0. \tag{2.57}$$

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., [29–31]).

3. Constructing ACGM

3.1 A Design Pattern for First-order Accelerated Algorithms

The augmented estimate sequence property (AESP) in (2.35) constitutes a sufficient condition for an algorithm to have the convergence guarantees at every iteration $k \geq 0$ given by A_k as in (2.6). However, it does not provide a means of computing iterates.

The design procedure we propose in this work is in line with the derivation found in [16]. We strive here for generality and rely, as much as possible, on fundamental arguments.

An optimization algorithm essentially employs at every iteration k a generator for $\boldsymbol{x}_{k+1}$. We devise this generator to ensure that $\boldsymbol{x}_{k+1}$ obeys the AESP *for any algorithmic state*. The AESP is an upper bound property of $F(\boldsymbol{x}_{k+1})$, which needs to be satisfied *before* $\boldsymbol{x}_{k+1}$ is computed. Therefore, $F(\boldsymbol{x}_{k+1})$ has to be substituted with a simple upper bound. Since we don't know *how* to compute $\boldsymbol{x}_{k+1}$ at this point, we need to define this upper bound as a function over the entire domain, which we denote as $u_{k+1}(\boldsymbol{x})$, that should be a *local upper bound* on the objective at $\boldsymbol{x}_{k+1}$, namely

$$F(\boldsymbol{x}_{k+1}) \leq u_{k+1}(\boldsymbol{x}_{k+1}), \quad k \geq 0. \tag{3.1}$$

We want to have the best convergence guarantees available. The ISDUB expression (2.6) implies that the values of $F(\boldsymbol{x}_{k+1})$ should be as low as possible. The simplest way to ensure this, and at the same time to provide a means of computing $\boldsymbol{x}_{k+1}$, is by setting $\boldsymbol{x}_{k+1}$ to be the optimal point of $u_{k+1}(\boldsymbol{x})$, that is

$$\boldsymbol{x}_{k+1} = \operatorname*{arg\,min}_{\boldsymbol{x} \in \mathbb{R}^n} u_{k+1}(\boldsymbol{x}), \quad k \geq 0. \tag{3.2}$$

Methods that employ this technique are often denoted as majorization minimization (MM) algorithms (see, e.g., [32, 33] and references therein).

The Lipschitz gradient property of f implies the existence, for a given control point $\boldsymbol{y}_{k+1} \in \mathbb{R}^n$, of a unique upper bound on $f(\boldsymbol{x})$ in the form of

$Q_{f,L_f,\boldsymbol{y}_{k+1}}(\boldsymbol{x})$ (see (2.43)). Regularizer Ψ is potentially unbounded above but, due to its simplicity, it is practical to consider it to be its own upper bound. Thus, an obvious choice for $u_{k+1}(\boldsymbol{x})$ would be

$$u_{k+1}(\boldsymbol{x}) = Q_{f,L_f,\boldsymbol{y}_{k+1}}(\boldsymbol{x}) + \Psi(\boldsymbol{x}), \quad k \geq 0. \tag{3.3}$$

The upper bound in (3.3) takes the form of a composite parabola whose optimum point can be readily obtained using the proximal operator. However, it depends on the global Lipschitz constant L_f, which poses two problems. First, this constant may not be known to the algorithm, either because it is intractable to compute or because a closed form expression of the smooth part f may not be available. Second, the path taken by the algorithm could traverse a region where the curvature of the objective is far lower than L_f would seem to indicate. Both problems can be alleviated by computing at every iteration k a Lipschitz constant estimate (LCE) that we denote by L_{k+1}. Therefore, the LCEs not only allow the algorithm to convergence when a value of L_f is not available, but may increase convergence speed even when L_f is known.

Although algorithmic prerequisites compel us to depart from the obvious choice in (3.3), upper bounds $u_{k+1}(\boldsymbol{x})$ can still be fully determined by $\boldsymbol{y}_{k+1}$ and L_{k+1} in the most common design scenarios, as we will show throughout this chapter.

Alongside the iterates, the algorithm must update the estimate sequence. The shape of the estimate functions is the same as their augmented counterparts and in the following we only consider the estimate sequence. Every new estimate function $\psi_{k+1}(\boldsymbol{x})$ contains a *global lower bound* term $W_{k+1}(\boldsymbol{x})$. The μ_f strong convexity of f and the μ_Ψ strong convexity of Ψ ensure the existence of a multitude of simple global lower bounds on F, each obtained by combining a simple (generalized parabolic) lower bound on f with a (generalized parabolic or Ψ itself) simple lower bound on Ψ. We denote the new simple lower bound at every iteration k by $w_{k+1}(\boldsymbol{x})$. Unlike $u_{k+1}(\boldsymbol{x})$, $w_{k+1}(\boldsymbol{x})$ may not be defined in terms of a single point due to the subdifferentiability of Ψ.

Setting the estimate function lower bound $W_{k+1}(\boldsymbol{x})$ to $w_{k+1}(\boldsymbol{x})$ would mean discarding the information revealed by past oracle calls used to compute $w_i(\boldsymbol{x})$ for $i \in \{1, ..., k\}$. Note that indexing starts at 1 instead of 0 because W_0 is not used and assumed null. The most computationally efficient means of incorporating past information in $W_{k+1}(\boldsymbol{x})$ is by weighted averaging all simple lower bounds computed by the algorithm up until iteration k. Among the weighting strategies, by far the least computationally demanding (as we shall demonstrate in the sequel) consists of

$$W_{k+1}(\boldsymbol{x}) = \frac{\sum_{i=1}^{k+1} a_i w_i(\boldsymbol{x})}{\sum_{i=1}^{k+1} a_i}, \quad \boldsymbol{x} \in \mathbb{R}^n, \tag{3.4}$$

where the weight at each iteration k, a_{k+1}, is given by

$$a_{k+1} = A_{k+1} - A_k, \quad k \geq 0. \tag{3.5}$$

The sequence of convergence guarantees is increasing which implies that $a_{k+1} > 0$ for all $k \geq 0$. Therefore, the weighting in (3.5) produces in (3.4) *valid* global lower bounds on the objective F.

The *minimalistic* quality of this strategy can also be easily shown. Applying the weighting described in (3.4) and (3.5) in (2.12) produces an estimate function expression comprising the initial estimate function plus a weighted sum of all previous lower bounds, that is

$$\psi_k(\boldsymbol{x}) = \sum_{i=1}^{k} a_i w_i(\boldsymbol{x}) + \psi_0(\boldsymbol{x}), \quad k \geq 1. \tag{3.6}$$

The estimate sequence update at iteration k simply becomes

$$\psi_{k+1}(\boldsymbol{x}) = \psi_k(\boldsymbol{x}) + a_{k+1} w_{k+1}(\boldsymbol{x}), \quad k \geq 0. \tag{3.7}$$

Convergence guarantees accumulate past weights and alleviate the need to store these weights individually across iterations. Thus, the state of the algorithm at the beginning of iteration k (the old state) is given by the iterate $\boldsymbol{x}_k$, estimate function ψ_k, convergence guarantee A_k, and LCE L_k. At every iteration, the algorithm needs to compute the auxiliary point y_{k+1} and the new LCE L_{k+1} to determine the upper bound $u_{k+1}(\boldsymbol{x})$ which, in turn, can be used to generate the new iterate $\boldsymbol{x}_{k+1}$. It also needs to calculate the new weight a_{k+1} and the global lower bound $w_{k+1}(\boldsymbol{x})$ to create a new estimate function ψ_{k+1} according to (3.7). The new convergence guarantee A_{k+1} is obtained according to (3.5). The bounds $u_{k+1}(\boldsymbol{x})$ and $w_{k+1}(\boldsymbol{x})$ may be correlated and are computed together at this stage. This basic structure of first-order algorithms for composite problems with a provable convergence rate is summarized in Algorithm 1.

Here, $\mathcal{C}$ is a process that computes the new parameters from the old state of the algorithm. The subscripts denote which parameters it outputs. Thus, the design process of a particular algorithm reduces to the derivation of $\mathcal{C}$.

Algorithm 1 A basic structure for first-order accelerated algorithms

for $k = 0, \dots, K-1$ **do**
  $L_{k+1}, a_{k+1}, u_{k+1}(\boldsymbol{x}), w_{k+1}(\boldsymbol{x}) = \mathcal{C}_{L,a,u,w}(\boldsymbol{x}_k, \psi_k, A_k, L_k)$
  $A_{k+1} = A_k + a_{k+1}$
  $\boldsymbol{x}_{k+1} = \arg\min_{\boldsymbol{x} \in \mathbb{R}^n} u_{k+1}(\boldsymbol{x})$
  $\psi_{k+1}(\boldsymbol{x}) = \psi_k(\boldsymbol{x}) + a_{k+1} w_{k+1}(\boldsymbol{x})$
end for

3.1.1 Line-search

An algorithm that overestimates the local Lipschitz constant remains valid, albeit slow, but underestimation may invalidate the convergence guarantees. Consequently, an algorithm must employ a *validation rule* $\mathcal{L}$ that determines whether a candidate LCE $\hat{L}_{k+1}$ is sufficiently large or not. A very simple estimation strategy consists of gradually decreasing candidates by multiplying once at the beginning of every iteration with $r_d < 1$ until an iteration is reached when the validation rule no longer holds. This step can be omitted for computational reasons, in which case we consider that $r_d = 1$. The candidate is then increased during that iteration by successively multiplying with $r_u > 1$ until the candidate passes the validation rule. In this context, the validation rule becomes a line-search stopping criterion (LSSC). The last valid candidate LCE is set as L_{k+1} and the search at the next iteration starts from there. The above process is often referred to in the literature as an Armijo-type [34] backtracking line-search strategy. Each increase of a candidate constitutes a "backtrack" because it results in a smaller step size.

Line-search involves only the upper bound $u_{k+1}(\boldsymbol{x})$ (uniquely determined by $\boldsymbol{y}_{k+1}$ and L_{k+1}) and the new iterate $\boldsymbol{x}_{k+1}$. Consequently, LSSC function $\mathcal{L}$ is parameterized by candidates $\hat{\boldsymbol{x}}_{k+1}$, $\hat{\boldsymbol{y}}_{k+1}$ and $\hat{L}_{k+1}$. The structure of a backtracking line-search first-order method is outlined in Algorithm 2.

Algorithm 2 takes as input the initial estimate function ψ_0 (parametrized by the starting point $\boldsymbol{x}_0 \in \mathbb{R}^n$, the initial weight $A_0 \geq 0$, and the initial curvature $\gamma_0 > 0$), the total number of iterations $K \geq 1$, and, if the Lipschitz constant is not known in advance, an initial estimate $L_0 > 0$.

3.2 Design Choices

We proceed with the design of ACGM based on the framework found in Algorithm 2. To maintain the augmented estimate sequence property (2.35), we enforce the stronger Lyapunov property of the gap sequence (2.57). However, the first and most important step in the design process is the selection of upper bounds $u_{k+1}(\boldsymbol{x})$ and lower bounds $w_{k+1}(\boldsymbol{x})$.

Algorithm 2 A basic structure for first-order accelerated algorithms employing backtracking line-search

for k = 0,...,K-1 **do**
  $\hat{L}_{k+1} := r_d L_k$
  loop
    $\hat{a}_{k+1}, \hat{u}_{k+1}(\boldsymbol{x}), \hat{w}_{k+1}(\boldsymbol{x}) := \mathcal{C}_{a,u,w}(\boldsymbol{x}_k, \psi_k, A_k, \hat{L}_{k+1})$
    $\hat{\boldsymbol{x}}_{k+1} := \underset{\boldsymbol{x}\in\mathbb{R}^n}{\arg\min}\, \hat{u}_{k+1}(\boldsymbol{x})$
    if $\mathcal{L}(\hat{\boldsymbol{x}}_{k+1}, \hat{\boldsymbol{y}}_{k+1}, \hat{L}_{k+1})$ **then**
      Break from loop
    else
      $\hat{L}_{k+1} := r_u \hat{L}_{k+1}$
    end if
  end loop
  $\boldsymbol{x}_{k+1} := \hat{\boldsymbol{x}}_{k+1},\ L_{k+1} := \hat{L}_{k+1},\ a_{k+1} := \hat{a}_{k+1}$
  $w_{k+1}(\boldsymbol{x}) := \hat{w}_{k+1}(\boldsymbol{x})$
  $A_{k+1} = A_k + a_{k+1}$
  $\psi_{k+1}(\boldsymbol{x}) = \psi_k(\boldsymbol{x}) + a_{k+1} w_{k+1}(\boldsymbol{x})$
end for

3.2.1 Upper Bounds

As we have seen in Section 3.1, the obvious choice of the upper bound in (3.3) has to account for the LCE. The resulting bound takes the form of

$$u_{k+1}(\boldsymbol{x}) = Q_{f,L_{k+1},\boldsymbol{y}_{k+1}}(\boldsymbol{x}) + \Psi(\boldsymbol{x}), \quad k \geq 0. \tag{3.8}$$

Note that even if the value of L_f is known beforehand, (3.3) may be a looser bound on F than (3.8) and may provide less information on the problem thereby slowing down the algorithm. Hence, we prefer (3.8) in the design of our algorithm.

Parabola $Q_{f,L_{k+1},\boldsymbol{y}_{k+1}}(\boldsymbol{x})$ can be written in canonical form (2.40) for all $k \geq 0$ as

$$\begin{aligned} Q_{f,L_{k+1},\boldsymbol{y}_{k+1}}(\boldsymbol{x}) &= f(\boldsymbol{y}_{k+1}) + \langle \nabla f(\boldsymbol{y}_{k+1}), \boldsymbol{x} - \boldsymbol{y}_{k+1} \rangle + \frac{L_{k+1}}{2} \|\boldsymbol{x} - \boldsymbol{y}_{k+1}\|_2^2 \\ &= f(\boldsymbol{y}_{k+1}) - \frac{1}{2L_{k+1}} \|\nabla f(\boldsymbol{y}_{k+1})\|_2^2 + \frac{L_{k+1}}{2} \|\boldsymbol{x} - \bar{\boldsymbol{y}}_{k+1}\|_2^2, \end{aligned} \tag{3.9}$$

where

$$\bar{\boldsymbol{y}}_{k+1} = \boldsymbol{y}_{k+1} - \frac{1}{L_{k+1}} \nabla f(\boldsymbol{y}_{k+1}), \quad k \geq 0. \tag{3.10}$$

The canonical form (3.9) can be used to express the optimum point of

$u_{k+1}(\boldsymbol{x})$ in (3.8). We have that

$$\begin{aligned}\operatorname*{arg\,min}_{\boldsymbol{x}\in\mathbb{R}^n} u_{k+1}(\boldsymbol{x}) &= \operatorname*{arg\,min}_{\boldsymbol{x}\in\mathbb{R}^n} \left(u_{k+1}(\boldsymbol{x}) - u^*_{k+1}\right) \\ &= \operatorname*{arg\,min}_{\boldsymbol{x}\in\mathbb{R}^n} \left(\Psi(\boldsymbol{x}) + \frac{L_{k+1}}{2}\left\|\boldsymbol{x} - \bar{\boldsymbol{y}}_{k+1}\right\|_2^2\right) \\ &= \operatorname{prox}_{\frac{1}{L_{k+1}}\Psi}(\bar{\boldsymbol{y}}_{k+1}), \quad k \geq 0. \end{aligned} \tag{3.11}$$

The iterate update in line 5 of Algorithm 2 becomes

$$\boldsymbol{x}_{k+1} = T_{f,\Psi,L_{k+1}}(\boldsymbol{y}_{k+1}), \quad k \geq 0, \tag{3.12}$$

where $T_{f,\Psi,L}(\boldsymbol{y})$ denotes the proximal gradient operator of the objective components f and Ψ using inverse step size $L > 0$, given by

$$T_{f,\Psi,L}(\boldsymbol{y}) = \operatorname{prox}_{\frac{1}{L}\Psi}\left(\boldsymbol{y} - \frac{1}{L}\nabla f(\boldsymbol{y})\right), \quad \boldsymbol{y} \in \mathbb{R}^n. \tag{3.13}$$

The LCE L_{k+1} must ensure that $u_{k+1}(\boldsymbol{x})$ is a local upper bound in the sense of (3.1). Combining (3.1) with (3.12) gives an LCE validation rule in the form of

$$F(\boldsymbol{x}_{k+1}) \leq Q_{f,L_{k+1},\boldsymbol{y}_{k+1}}(\boldsymbol{x}_{k+1}) + \Psi(\boldsymbol{x}_{k+1}), \quad k \geq 0. \tag{3.14}$$

The value $\Psi(\boldsymbol{x}_{k+1})$ appears on both sides of the equation. By subtracting it we obtain the descent condition for f, written as

$$f(\boldsymbol{x}_{k+1}) \leq Q_{f,L_{k+1},\boldsymbol{y}_{k+1}}(\boldsymbol{x}_{k+1}), \quad k \geq 0. \tag{3.15}$$

Thus, the derivation of the upper bounds has given us not only an iterate update rule that yields $\boldsymbol{x}_{k+1}$ but also a Lipschitz constant estimation rule for L_{k+1}. Both quantities will play a role in the derivation of the *lower* bounds as well.

Composite Gradient

To further simplify notation when dealing with the lower bounds and to compare our method more easily to Nesterov's FGM, we use Nesterov's *composite gradient* [17], defined as

$$g_L(\boldsymbol{y}) \stackrel{\text{def}}{=} L(\boldsymbol{y} - \operatorname*{arg\,min}_{\boldsymbol{x}\in\mathbb{R}^n}(Q_{f,L,\boldsymbol{y}}(\boldsymbol{x}) + \Psi(\boldsymbol{x})), \quad \boldsymbol{y} \in \mathbb{R}^n, \quad L > 0. \tag{3.16}$$

We shall see later on that, in this work, the composite gradient need not be parametrized by f and Ψ.

In FGM, new iterates are given by a gradient descent step as

$$\boldsymbol{x}_{k+1} = \boldsymbol{y}_{k+1} - \frac{1}{L_f}\nabla f(\boldsymbol{y}_{k+1}), \quad k \geq 0. \tag{3.17}$$

The composite gradient is therefore defined as a substitute for the gradient in the iterate update (3.12), which becomes

$$\boldsymbol{x}_{k+1} = \boldsymbol{y}_{k+1} - \frac{1}{L_{k+1}} g_{L_{k+1}}(\boldsymbol{y}_{k+1}), \quad k \geq 0. \tag{3.18}$$

The new iterate $\boldsymbol{x}_{k+1}$ satisfies the first-order optimality condition of local upper bounds $u_{k+1}(\boldsymbol{x})$ in (3.8). This can be expressed as the existence of a subgradient $\boldsymbol{\xi}_{k+1}$ of regularizer Ψ at $\boldsymbol{x}_{k+1}$ such that

$$\nabla_{\boldsymbol{x}} Q_{f, L_{k+1}, \boldsymbol{y}_{k+1}}(\boldsymbol{x}_{k+1}) + \boldsymbol{\xi}_{k+1} = 0, \quad k \geq 0. \tag{3.19}$$

Expanding the gradient over $\boldsymbol{x}$ term gives

$$\nabla f(\boldsymbol{y}_{k+1}) + L_{k+1}(\boldsymbol{x}_{k+1} - \boldsymbol{y}_{k+1}) + \boldsymbol{\xi}_{k+1} = 0, \quad k \geq 0. \tag{3.20}$$

Therefore $\boldsymbol{\xi}_{k+1}$ is uniquely determined by

$$\boldsymbol{\xi}_{k+1} = L_{k+1}(\boldsymbol{y}_{k+1} - \boldsymbol{x}_{k+1}) - \nabla f(\boldsymbol{y}_{k+1}), \quad k \geq 0. \tag{3.21}$$

Substituting (3.21) in the composite gradient iterate update (3.18) yields

$$g_{L_{k+1}}(\boldsymbol{y}_{k+1}) = \nabla f(\boldsymbol{y}_{k+1}) + \boldsymbol{\xi}_{k+1}, \quad k \geq 0. \tag{3.22}$$

The form in (3.22) explains the origin of the term composite gradient, as the sum of the gradient of f at $\boldsymbol{y}_{k+1}$ and a particular subgradient of Ψ at $\boldsymbol{x}_{k+1}$.

3.3 ACGM for Non-strongly Convex Objectives

For simplicity of exposition, we first construct our method for the non-strongly convex case ($\mu = \mu_f = \mu_\Psi = 0$) and we later generalize the analysis to arbitrary strong convexity.

3.3.1 Lower Bounds

The generation of $\boldsymbol{x}_{k+1}$ during iteration $k \geq 0$ involves two oracle calls: a call to ∇f at $\boldsymbol{y}_{k+1}$ and to $prox_{\tau\Psi}$ with step size $\tau = 1/L_{k+1}$ at point $\bar{\boldsymbol{y}}_{k+1}$. As shown in (3.22), this proximal gradient call yields $\nabla f(\boldsymbol{y}_{k+1})$ and $\boldsymbol{\xi}_{k+1} \in \partial\Psi(\boldsymbol{x}_{k+1})$. We aim to keep the number of oracle calls in our algorithm to a minimum and we reuse these quantities when constructing the lower bounds.

The convexity of f yields

$$f(\boldsymbol{x}) \geq f(\boldsymbol{y}_{k+1}) + \langle \nabla f(\boldsymbol{y}_{k+1}), \boldsymbol{x} - \boldsymbol{y}_{k+1} \rangle, \quad \boldsymbol{x} \in \mathbb{R}^n, \quad k \geq 0 \tag{3.23}$$

and the subdifferentiability of Ψ gives

$$\begin{aligned} \Psi(\boldsymbol{x}) &\geq \Psi(\boldsymbol{x}_{k+1}) + \langle \boldsymbol{\xi}_{k+1}, \boldsymbol{x} - \boldsymbol{x}_{k+1} \rangle \\ &= \Psi(\boldsymbol{x}_{k+1}) + \langle \boldsymbol{\xi}_{k+1}, \boldsymbol{x} - \boldsymbol{y}_{k+1} \rangle + \langle \boldsymbol{\xi}_{k+1}, \boldsymbol{y}_{k+1} - \boldsymbol{x}_{k+1} \rangle, \quad \boldsymbol{x} \in \mathbb{R}^n, \quad k \geq 0. \end{aligned} \tag{3.24}$$

Adding (3.23) and (3.24) together we obtain

$$\begin{aligned} F(\boldsymbol{x}) \geq f(\boldsymbol{y}_{k+1}) + \Psi(\boldsymbol{x}_{k+1}) + \langle \boldsymbol{\xi}_{k+1}, \boldsymbol{y}_{k+1} - \boldsymbol{x}_{k+1} \rangle \\ + \left\langle g_{L_{k+1}}(\boldsymbol{y}_{k+1}), \boldsymbol{x} - \boldsymbol{y}_{k+1} \right\rangle, \quad \boldsymbol{x} \in \mathbb{R}^n, \quad k \geq 0. \end{aligned} \tag{3.25}$$

The right-hand side (RHS) of (3.25) is a global lower bound on F. However, the augmented estimate sequence property (2.35) and the local upper bound property (3.1) are based on $F(\boldsymbol{x}_{k+1})$ whereas (3.25) contains $f(\boldsymbol{y}_{k+1})$. The descent condition in (3.15) means that the line search residual, given by

$$\begin{aligned} \mathcal{N}_{k+1} &\stackrel{\text{def}}{=} Q_{f,L_{k+1},\boldsymbol{y}_{k+1}}(\boldsymbol{x}_{k+1}) - f(\boldsymbol{x}_{k+1}) \\ &= f(\boldsymbol{y}_{k+1}) - f(\boldsymbol{x}_{k+1}) + \langle \nabla f(\boldsymbol{y}_{k+1}), \boldsymbol{x}_{k+1} - \boldsymbol{y}_{k+1} \rangle \\ &\quad + \frac{L_{k+1}}{2} \|\boldsymbol{x}_{k+1} - \boldsymbol{y}_{k+1}\|_2^2, \quad k \geq 0, \end{aligned} \tag{3.26}$$

is non-negative. Furthermore, the backtracking line search procedure is designed to reduce this residual as much as possible. Therefore, we can subtract it from the RHS of (3.25) without significantly decreasing the tightness of the lower bound. We thus obtain

$$\begin{aligned} F(\boldsymbol{x}) \geq F(\boldsymbol{x}_{k+1}) + \langle \boldsymbol{\xi}_{k+1} + \nabla f(\boldsymbol{y}_{k+1}), \boldsymbol{y}_{k+1} - \boldsymbol{x}_{k+1} \rangle - \frac{L_{k+1}}{2} \|\boldsymbol{x}_{k+1} - \boldsymbol{y}_{k+1}\|_2^2 \\ + \left\langle g_{L_{k+1}}(\boldsymbol{y}_{k+1}), \boldsymbol{x} - \boldsymbol{y}_{k+1} \right\rangle, \quad \boldsymbol{x} \in \mathbb{R}^n, \quad k \geq 0. \end{aligned} \tag{3.27}$$

The difference between points $\boldsymbol{y}_{k+1}$ and $\boldsymbol{x}_{k+1}$ can be expressed using the composite gradient. Subgradient expression (3.21) combined with (3.22) gives

$$g_{L_{k+1}}(\boldsymbol{y}_{k+1}) = L_{k+1}(\boldsymbol{y}_{k+1} - \boldsymbol{x}_{k+1}), \quad k \geq 0. \tag{3.28}$$

Applying (3.28) and (3.22) in (3.27) produces a simple lower bound, given by

$$\begin{aligned} F(\boldsymbol{x}) \geq F(\boldsymbol{x}_{k+1}) + \frac{1}{2L_{k+1}} \|g_{L_{k+1}}(\boldsymbol{y}_{k+1})\|_2^2 \\ + \left\langle g_{L_{k+1}}(\boldsymbol{y}_{k+1}), \boldsymbol{x} - \boldsymbol{y}_{k+1} \right\rangle, \quad \boldsymbol{x} \in \mathbb{R}^n, \quad k \geq 0, \end{aligned} \tag{3.29}$$

which we denote as the relaxed supporting hyperplane property.

We have chosen auxiliary point $\boldsymbol{y}_{k+1}$ as the support in (3.24) for convenience. Choosing $\boldsymbol{x}_{k+1}$ instead does not alter the analysis, since the difference between the two points can be expressed using (3.28).

3.3.2 Formulating ACGM

We proceed with the design of our method, ACGM, based on the basic structure for first-order accelerated algorithms employing backtracking line-search presented in Algorithm 2. The building blocks are:

1. The composite parabolic upper bounds in (3.8).

2. The relaxed supporting hyperplane lower bounds from (3.29), written as

$$\begin{aligned} w_{k+1}(\boldsymbol{x}) = F(\boldsymbol{x}_{k+1}) + \frac{1}{2L_{k+1}}\|g_{L_{k+1}}(\boldsymbol{y}_{k+1})\|_2^2 \\ + \left\langle g_{L_{k+1}}(\boldsymbol{y}_{k+1}), \boldsymbol{x} - \boldsymbol{y}_{k+1}\right\rangle, \quad \boldsymbol{x} \in \mathbb{R}^n, \quad k \geq 0. \end{aligned} \tag{3.30}$$

3. The Lyapunov property of the gap sequence in (2.57) .

The structure of the upper bounds implies that line 5 in Algorithm 2 is the proximal gradient step (3.12). The local upper bound property of $u_{k+1}(\boldsymbol{x})$ in (3.8) also gives a line-search stopping criterion (line 6 in Algorithm 2) in the form of (3.15).

The lower bounds are hyperplanes. The weighting strategy (3.4) ensures that all estimate function lower bounds are hyperplanes as well. Then, for all $k \geq 0$, the estimate function ψ_k along with its augmented variant ψ_k' are parabolae with the curvature given by γ_0, written as

$$\psi_k(\boldsymbol{x}) = \psi_k^* + \frac{\gamma_0}{2}\|\boldsymbol{x} - \boldsymbol{v}_k\|_2^2, \qquad \psi_k'(\boldsymbol{x}) = \psi_k'^* + \frac{\gamma_0}{2}\|\boldsymbol{x} - \boldsymbol{v}_k\|_2^2. \tag{3.31}$$

Since estimate functions can be written in canonical parabolic form, differentiating with respect to $\boldsymbol{x}$ the estimate sequence update in line 15 of Algorithm 2 using lower bound (3.30) results in

$$\gamma_0(\boldsymbol{x} - \boldsymbol{v}_{k+1}) = \gamma_0(\boldsymbol{x} - \boldsymbol{v}_{k+1}) + a_{k+1}g_{L_{k+1}}(\boldsymbol{y}_{k+1}), \quad \boldsymbol{x} \in \mathbb{R}^n, \quad k \geq 0. \tag{3.32}$$

This gives a vertex update rule in the form of

$$\boldsymbol{v}_{k+1} = \boldsymbol{v}_k - \frac{a_{k+1}}{\gamma_0}g_{L_{k+1}}(\boldsymbol{y}_{k+1}), \quad k \geq 0. \tag{3.33}$$

Next, we devise update rules for a_{k+1} and $\boldsymbol{y}_{k+1}$ in ACGM such that the Lyapunov property (2.57) is guaranteed to hold for every $k \geq 0$ and for any algorithmic state. The following result provides a means of accomplishing this.

Theorem 3. *If at iteration $k \geq 0$, the descent condition for f in* (3.15) *holds, then*

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

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

$$\begin{aligned} \mathcal{A}_{k+1} &\stackrel{\text{def}}{=} \frac{1}{2}\left(\frac{A_{k+1}}{L_{k+1}} - \frac{a_{k+1}^2}{\gamma_0}\right)\|g_{L_{k+1}}(\boldsymbol{y}_{k+1})\|_2^2, \\ \mathcal{B}_{k+1} &\stackrel{\text{def}}{=} \left\langle g_{L_{k+1}}(\boldsymbol{y}_{k+1}), A_k\boldsymbol{x}_k + a_{k+1}\boldsymbol{v}_k - A_{k+1}\boldsymbol{y}_{k+1}\right\rangle. \end{aligned}$$

Proof. All the definitions and results within the scope of this proof apply for any $k \geq 0$. Let the tightness of the lower bound $w_{k+1}(\boldsymbol{x})$ at $\boldsymbol{x}$ be denoted by the residual $R_{k+1}(\boldsymbol{x})$ as

$$\begin{aligned} R_{k+1}(\boldsymbol{x}) \stackrel{\text{def}}{=} F(\boldsymbol{x}) - F(\boldsymbol{x}_{k+1}) - \frac{1}{2L_{k+1}}\|g_{L_{k+1}}(\boldsymbol{y}_{k+1})\|_2^2 \\ - \left\langle g_{L_{k+1}}(\boldsymbol{y}_{k+1}), \boldsymbol{x} - \boldsymbol{y}_{k+1}\right\rangle, \quad \boldsymbol{x} \in \mathbb{R}^n. \end{aligned} \tag{3.34}$$

From (3.29) we have that $R_{k+1}(\boldsymbol{x}) \geq 0$ for all $\boldsymbol{x} \in \mathbb{R}^n$. Therefore

$$A_k R_{k+1}(\boldsymbol{x}_k) + a_{k+1} R_{k+1}(\boldsymbol{x}^*) \geq 0. \tag{3.35}$$

By expanding terms we obtain

$$\begin{aligned} A_k F(\boldsymbol{x}_k) + a_{k+1} F(\boldsymbol{x}^*) - A_{k+1} F(\boldsymbol{x}_{k+1}) - \frac{A_{k+1}}{2L_{k+1}} \|g_{L_{k+1}}(\boldsymbol{y}_{k+1})\|_2^2 \\ - \left\langle g_{L_{k+1}}(\boldsymbol{y}_{k+1}), A_k \boldsymbol{x}_k + a_{k+1} \boldsymbol{x}^* - A_{k+1} \boldsymbol{y}_{k+1} \right\rangle \geq 0. \end{aligned} \tag{3.36}$$

This is equivalent to

$$\begin{aligned} A_k (F(\boldsymbol{x}_k) - F(\boldsymbol{x}^*)) - A_{k+1} (F(\boldsymbol{x}_{k+1}) - F(\boldsymbol{x}^*)) \geq \frac{A_{k+1}}{2L_{k+1}} \|g_{L_{k+1}}(\boldsymbol{y}_{k+1})\|_2^2 \\ + \left\langle g_{L_{k+1}}(\boldsymbol{y}_{k+1}), A_k \boldsymbol{x}_k + a_{k+1} \boldsymbol{x}^* - A_{k+1} \boldsymbol{y}_{k+1} \right\rangle. \end{aligned} \tag{3.37}$$

To isolate the gap sequence terms, we add to both sides of (3.37) the quantity

$$V_{k+1} \overset{\text{def}}{=} \frac{\gamma_0}{2} \|\boldsymbol{v}_k - \boldsymbol{x}^*\|_2^2 - \frac{\gamma_0}{2} \|\boldsymbol{v}_{k+1} - \boldsymbol{x}^*\|_2^2 \tag{3.38}$$

to obtain

$$\begin{aligned} \Delta_{k+1} + \frac{A_{k+1}}{2L_{k+1}} \|g_{L_{k+1}}(\boldsymbol{y}_{k+1})\|_2^2 \\ + \left\langle g_{L_{k+1}}(\boldsymbol{y}_{k+1}), A_k \boldsymbol{x}_k + a_{k+1} \boldsymbol{x}^* - A_{k+1} \boldsymbol{y}_{k+1} \right\rangle + V_{k+1} \leq \Delta_k. \end{aligned} \tag{3.39}$$

Using (3.33) in (3.38) we obtain

$$\begin{aligned} V_{k+1} &= \frac{\gamma_0}{2} \left(\|\boldsymbol{v}_k\|_2 - \|\boldsymbol{v}_{k+1}\|_2 \right) + \gamma_0 \left\langle \boldsymbol{v}_{k+1} - \boldsymbol{v}_k, \boldsymbol{x}^* \right\rangle \\ &= \frac{\gamma_0}{2} \left\langle \boldsymbol{v}_k - \boldsymbol{v}_{k+1}, \boldsymbol{v}_k + \boldsymbol{v}_{k+1} \right\rangle - \left\langle g_{L_{k+1}}(\boldsymbol{y}_{k+1}), a_{k+1} \boldsymbol{x}^* \right\rangle \\ &= \frac{1}{2} \left\langle a_{k+1} g_{L_{k+1}}(\boldsymbol{y}_{k+1}), 2\boldsymbol{v}_k - \frac{a_{k+1}}{\gamma_0} g_{L_{k+1}}(\boldsymbol{y}_{k+1}) \right\rangle - \left\langle g_{L_{k+1}}(\boldsymbol{y}_{k+1}), a_{k+1} \boldsymbol{x}^* \right\rangle \\ &= -\frac{a_{k+1}}{2\gamma_0} \|g_{L_{k+1}}(\boldsymbol{y}_{k+1})\|_2^2 + \left\langle g_{L_{k+1}}(\boldsymbol{y}_{k+1}), a_{k+1} \boldsymbol{v}_k - a_{k+1} \boldsymbol{x}^* \right\rangle. \end{aligned} \tag{3.40}$$

Substituting (3.40) in (3.39) and rearranging terms completes the proof. □

A sufficient condition for (2.57) to hold is to have $\mathcal{A}_{k+1} \geq 0$ and $\mathcal{B}_{k+1} \geq 0$ for every $k \geq 0$. Since $\|g_{L_{k+1}}(\boldsymbol{y}_{k+1})\|_2^2$ is a non-negative quantity, $\mathcal{A}_{k+1} \geq 0$ can be ensured for any possible algorithmic state if

$$A_{k+1} \gamma_0 \geq L_{k+1} a_{k+1}^2, \quad k \geq 0. \tag{3.41}$$

However, the angle between $A_k \boldsymbol{x}_k + a_{k+1} \boldsymbol{v}_k$ and $g_{L_{k+1}}(\boldsymbol{y}_{k+1})$ may be obtuse, meaning that $\mathcal{B}_{k+1}$ may be non-positive. Again, to account for this possibility without additional oracle queries, we impose $\mathcal{B}_{k+1} = 0$ by setting auxiliary point $\boldsymbol{y}_{k+1}$ to be

$$\boldsymbol{y}_{k+1} = \frac{1}{A_{k+1}} \left(A_k \boldsymbol{x}_k + a_{k+1} \boldsymbol{v}_k \right), \quad k \geq 0. \tag{3.42}$$

Scale γ_0 invariance
The weight condition in (3.41) as well as the auxiliary point update in (3.42) and the vertex update in (3.33) are, respectively, equivalent for all $k \geq 0$ to

$$\frac{A_{k+1}}{\gamma_0} \geq L_{k+1} \left(\frac{a_{k+1}}{\gamma_0} \right)^2, \tag{3.43}$$

$$\boldsymbol{y}_{k+1} = \frac{1}{\frac{A_{k+1}}{\gamma_0}} \left(\frac{A_k}{\gamma_0} \boldsymbol{x}_k + \frac{a_{k+1}}{\gamma_0} \boldsymbol{v}_k \right), \tag{3.44}$$

$$\boldsymbol{v}_{k+1} = \boldsymbol{v}_k - \frac{a_{k+1}}{\gamma_0} g_{L_{k+1}}(\boldsymbol{y}_{k+1}). \tag{3.45}$$

In addition, the Lipschitz constant estimation procedure, whereby the validation rule corresponds to the descent condition in (3.15), and the composite gradient expression in (3.28) do not depend on the weights and convergence guarantees. Thus the behavior of the algorithm is not affected by the scale of γ_0. To reduce the overall computation cost, we set γ_0 to 1.

To obtain the largest increase in the convergence guarantees, we enforce equality in (3.41), namely

$$L_{k+1} a_{k+1}^2 = A_{k+1}, \quad k \geq 0. \tag{3.46}$$

The equality in (3.46) can be recast as a quadratic equation in a_{k+1} in the form of

$$L_{k+1} a_{k+1}^2 - a_{k+1} - A_k = 0, \quad k \geq 0. \tag{3.47}$$

Given that $L_{k+1} > 0$ and $A_k \geq 0$, (3.47) has only one positive root a_{k+1}. The closed form expression of this root constitutes a weight update rule, given by

$$a_{k+1} = \frac{1 + \sqrt{1 + 4 L_{k+1} A_k}}{2 L_{k+1}}, \quad k \geq 0. \tag{3.48}$$

There is no need to compute the composite gradient explicitly. Instead, by using the definition of the composite gradient in (3.18), the update rule for the augmented estimate sequence vertices in (3.33) can be written in terms of the iterates and the auxiliary points as

$$\boldsymbol{v}_{k+1} = \boldsymbol{v}_k + a_{k+1} L_{k+1} (\boldsymbol{x}_{k+1} - \boldsymbol{y}_{k+1}), \quad k \geq 0. \tag{3.49}$$

The weight update in (3.48), the auxiliary point update in (3.42), and the upper bound expression in (3.8) leading to the iterate update in (3.12) make up the parameter generation procedure $\mathcal{C}_{a,u,w}$ in the design pattern given by Algorithm 2. By replacing them accordingly in Algorithm 2, along with the LSSC in (3.15) and the vertex update in (3.49), we obtain the ACGM algorithm for non-strongly convex objectives listed in Algorithm 3.

Algorithm 3 ACGM for non-strongly convex objectives in estimate sequence form

ACGM$(\boldsymbol{x}_0, L_0, A_0, r_u, r_d, K)$

$\boldsymbol{v}_0 = \boldsymbol{x}_0$
for $k = 0, \dots, K-1$ **do**
  $\hat{L}_{k+1} := r_d L_k$
  loop
    $\hat{a}_{k+1} := \frac{1+\sqrt{1+4\hat{L}_{k+1}A_k}}{2\hat{L}_{k+1}}$
    $\hat{A}_{k+1} := A_k + \hat{a}_{k+1}$
    $\hat{\boldsymbol{y}}_{k+1} := \frac{1}{\hat{A}_{k+1}}\left(A_k \boldsymbol{x}_k + \hat{a}_{k+1}\boldsymbol{v}_k\right)$
    $\hat{\boldsymbol{x}}_{k+1} := \mathrm{prox}_{\frac{1}{\hat{L}_{k+1}}\Psi}\left(\hat{\boldsymbol{y}}_{k+1} - \frac{1}{\hat{L}_{k+1}}\nabla f(\hat{\boldsymbol{y}}_{k+1})\right)$
    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
  $L_{k+1} = \hat{L}_{k+1},\ a_{k+1} = \hat{a}_{k+1},\ A_{k+1} = \hat{A}_{k+1}$
  $\boldsymbol{x}_{k+1} = \hat{\boldsymbol{x}}_{k+1},\ \boldsymbol{y}_{k+1} = \hat{\boldsymbol{y}}_{k+1}$
  $\boldsymbol{v}_{k+1} = \boldsymbol{v}_k + a_{k+1}L_{k+1}(\boldsymbol{x}_{k+1} - \boldsymbol{y}_{k+1})$
end for

3.3.3 Extrapolated Form

ACGM exhibits an interesting property that alleviates the need to explicitly maintain the vertices.

Lemma 3. *The estimate sequence vertices can be obtained from the main iterates through extrapolation, namely*

$$\boldsymbol{v}_{k+1} = \boldsymbol{x}_k + \frac{A_{k+1}}{a_{k+1}}(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k), \quad k \geq 0. \tag{3.50}$$

Proof. For ease of exposition, we accompany our analytical approach with arguments from Euclidean geometry.

In the first iteration ($k = 0$), we have that $\boldsymbol{v}_0 = \boldsymbol{x}_0$. Since auxiliary point $\boldsymbol{y}_1$ is a convex combination between $\boldsymbol{v}_0$ and $\boldsymbol{x}_0$, it is also equal to $\boldsymbol{x}_0$. The composite gradient $g_{L_1}(\boldsymbol{y}_1)$ then becomes

$$g_{L_1}(\boldsymbol{y}_1) = L_1(\boldsymbol{y}_1 - \boldsymbol{x}_1) = -L_1(\boldsymbol{x}_1 - \boldsymbol{x}_0). \tag{3.51}$$

Vertex $\boldsymbol{v}_1$ is obtained from $\boldsymbol{v}_0 = \boldsymbol{x}_0$ by adding a multiple of the composite gradient $g_{L_1}(\boldsymbol{y}_1)$, as

$$\boldsymbol{v}_1 = \boldsymbol{v}_0 - a_1 g_{L_1}(\boldsymbol{y}_1) \overset{(3.51)}{=} \boldsymbol{x}_0 + a_1 L_1(\boldsymbol{x}_1 - \boldsymbol{y}_1) \overset{(3.46)}{=} \boldsymbol{x}_1 + \frac{A_1}{a_1}(\boldsymbol{x}_1 - \boldsymbol{x}_0). \tag{3.52}$$

Figure 3.1(a) provides a visual proof of (3.52).

For subsequent ($k \geq 1$) iterations, we have that

$$\boldsymbol{x}_{k+1} - \boldsymbol{y}_{k+1} = -\frac{1}{L_{k+1}} g_{L_{k+1}}(\boldsymbol{y}_{k+1}) \overset{(3.46)}{=} -\frac{a_{k+1}^2}{A_{k+1}} g_{L_{k+1}}(\boldsymbol{y}_{k+1}) \tag{3.53}$$

$$\boldsymbol{v}_{k+1} - \boldsymbol{v}_k = -a_{k+1} g_{L_{k+1}}(\boldsymbol{y}_{k+1}) \tag{3.54}$$

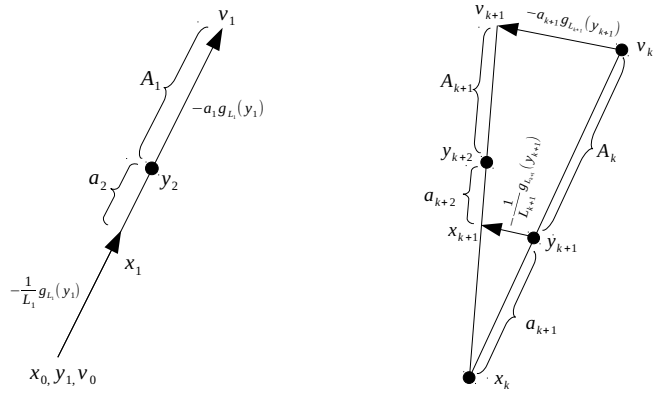
Therefore

$$\boldsymbol{v}_{k+1} - \boldsymbol{v}_k = \frac{A_{k+1}}{a_{k+1}}(\boldsymbol{x}_{k+1} - \boldsymbol{y}_{k+1}). \tag{3.55}$$

From (3.42) we also have that

$$\boldsymbol{v}_k - \boldsymbol{x}_k = \frac{A_{k+1}}{a_{k+1}}(\boldsymbol{y}_{k+1} - \boldsymbol{x}_k). \tag{3.56}$$

Adding together (3.55) and (3.56), we obtain the desired result. The extrapolation rule also follows from the parallelism and proportion in (3.55) combined with the collinearity and proportion in (3.56) using the Thales intercept theorem, as shown in Figure 3.1(b). □



(a) First iteration $k = 0$ (b) Subsequent iterations $k \geq 1$

Figure 3.1. Collinearity of $\boldsymbol{y}_{k+2}$, $\boldsymbol{v}_{k+1}$, $\boldsymbol{x}_{k+1}$, and $\boldsymbol{x}_k$ follows from the Thales intercept theorem

To bring our notation closer to that of FISTA and FISTA-CP, we define the following sequence $\{t_k\}_{k\geq 0}$ as

$$t_k \overset{\text{def}}{=} \sqrt{L_k A_k}, \quad k \geq 0. \tag{3.57}$$

For $k \geq 0$ we have that

$$t_{k+1} = \sqrt{L_{k+1} A_{k+1}} \overset{(3.46)}{=} \sqrt{\frac{A_{k+1}}{a_{k+1}^2} A_{k+1}} = \frac{A_{k+1}}{a_{k+1}}. \tag{3.58}$$

Therefore, t_{k+1} is the vertex extrapolation factor in Lemma 3.

The extrapolation property in Lemma 3 extends to the auxiliary point $\boldsymbol{y}_{k+1}$. To be able to use extrapolation in the first iteration, we further define $\boldsymbol{x}_{-1} \stackrel{\text{def}}{=} \boldsymbol{x}_0$. Using sequence $\{t_k\}_{k\geq 0}$, we can express this property succinctly.

Proposition 1. *At every iteration $k \geq 0$, the new auxiliary point y_{k+1} can be obtained from the previous two iterates through extrapolation as*

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

where

$$\beta_k = \frac{t_k - 1}{t_{k+1}}.$$

Proof. For $k = 0$ we have $\boldsymbol{x}_0 = \boldsymbol{x}_{-1}$ so

$$\boldsymbol{y}_1 = \frac{A_0\boldsymbol{x}_0 + a_1\boldsymbol{v}_0}{A_1} = \frac{A_0\boldsymbol{x}_0 + a_1\boldsymbol{x}_0}{A_1} = \boldsymbol{x}_0 = \boldsymbol{x}_0 + \frac{t_0 - 1}{t_1}(\boldsymbol{x}_0 - \boldsymbol{x}_{-1}). \quad (3.59)$$

Note that β_0 can take any real value. We choose $\beta_0 = \frac{t_0-1}{t_1}$ merely to have the same form for all $k \geq 0$.

For $k \geq 1$ we have

$$\boldsymbol{y}_{k+1} = \boldsymbol{x}_k + \frac{a_{k+1}}{A_{k+1}}(\boldsymbol{v}_k - \boldsymbol{x}_k). \quad (3.60)$$

Applying Lemma 3 at $k - 1$ we obtain

$$\begin{aligned}\boldsymbol{y}_{k+1} &= \boldsymbol{x}_k + \frac{a_{k+1}}{A_{k+1}}(\boldsymbol{x}_{k-1} + \frac{A_k}{a_k}(\boldsymbol{x}_k - \boldsymbol{x}_{k-1}) - \boldsymbol{x}_k) \\ &= \boldsymbol{x}_k + \frac{t_k(\boldsymbol{x}_k - \boldsymbol{x}_{k-1}) - (\boldsymbol{x}_k - \boldsymbol{x}_{k-1})}{t_{k+1}} \\ &= \boldsymbol{x}_k + \frac{t_k - 1}{t_{k+1}}(\boldsymbol{x}_k - \boldsymbol{x}_{k-1}). \quad (3.61)\end{aligned}$$

A simple qualitative visual proof of the extrapolation property can also be found in Figure 3.1(a) for $\boldsymbol{y}_2$ and in Figure 3.1(b) for $\boldsymbol{y}_k$ when $k \geq 3$. □

From definition (3.57) it is clear that for every $k \geq 0$ it is not necessary to maintain both A_k and t_k simultaneously. Therefore, we can derive a recursion rule for t_k independently of A_k.

Proposition 2. *The vertex extrapolation factors obey the following recursion rule:*

$$t_{k+1}^2 - t_{k+1} - \frac{L_{k+1}}{L_k}t_k^2 = 0, \quad k \geq 0.$$

Proof. First, based on (3.57) and (3.46) we derive the following expressions for all $k \geq 0$:

$$L_k A_k = t_k^2, \quad (3.62)$$

$$L_{k+1}a_{k+1} = \frac{A_{k+1}}{a_{k+1}} = t_{k+1}. \quad (3.63)$$

The convergence guarantee recursion rule (3.5) can be written as

$$A_{k+1} - a_{k+1} - A_k = 0, \quad k \geq 0. \tag{3.64}$$

Since all LCEs are positive we can write the above as

$$L_{k+1}A_{k+1} - L_{k+1}a_{k+1} - \frac{L_{k+1}}{L_k}L_k A_k = 0, \quad k \geq 0. \tag{3.65}$$

Substituting (3.62) and (3.63) in (3.65) completes the proof. □

From the definition in (3.57), we have that $t_{k+1} > 0$ for all $k \geq 0$. The quadratic equation in Proposition 2 has only one positive root, whose expression gives an update rule for t_{k+1} in the form of

$$t_{k+1} = \frac{1 + \sqrt{1 + 4\frac{L_{k+1}}{L_k}t_k^2}}{2}, \quad k \geq 0. \tag{3.66}$$

Based on Propositions 1 and 2, we can formulate ACGM for non-strongly convex objectives based on extrapolation, as listed in Algorithm 4. Algorithms 3 and 4 differ in form but are theoretically guaranteed to produce identical iterates.

Since convergence guarantee A_k is not longer maintained explicitly, it can be obtained from t_k as

$$A_k = \frac{t_k^2}{L_k}, \quad k \geq 0. \tag{3.67}$$

3.4 ACGM for Objectives with Arbitrary Strong Convexity

We continue the design of ACGM, this time for any $0 \leq \mu_f < L_f$ and any $\mu_\Psi \geq 0$. The design process follows closely the one previously shown in Section 3.3. While strong convexity poses new challenges, most of the properties derived in Section 3.3 carry over to this scenario.

3.4.1 Strong Convexity Transfer

To simplify the derivations, we transfer all strong convexity from Ψ to f, as

$$f'(\boldsymbol{x}) = f(\boldsymbol{x}) + \frac{\mu_\Psi}{2}\|\boldsymbol{x} - \boldsymbol{x}_0\|_2^2, \tag{3.68}$$

$$\Psi'(\boldsymbol{x}) = \Psi(\boldsymbol{x}) - \frac{\mu_\Psi}{2}\|\boldsymbol{x} - \boldsymbol{x}_0\|_2^2, \tag{3.69}$$

for all $\boldsymbol{x} \in \mathbb{R}^n$. As we shall see later, the center of strong convexity can be any point in $\mathbb{R}^n$. We choose $\boldsymbol{x}_0$ only for convenience. Likewise, the analysis can be performed with strong convexity transfered from f to Ψ. However, (3.68) implies that f' has Lipschitz gradient with constant $L_{f'} = L_f + \mu_\Psi$

Algorithm 4 ACGM for non-strongly convex objectives based on extrapolation

ACGM$(\boldsymbol{x}_0, L_0, A_0, r_u, r_d, K)$

$\boldsymbol{x}_{-1} = \boldsymbol{x}_0$
$t_0 = \sqrt{L_0 A_0}$
for $k = 0, ..., K-1$ **do**
  $\hat{L}_{k+1} := r_d L_k$
  loop
    $\hat{t}_{k+1} := \frac{1+\sqrt{1+4\frac{\hat{L}_{k+1}}{L_k}t_k^2}}{2}$
    $\hat{\boldsymbol{y}}_{k+1} := \boldsymbol{x}_k + \frac{t_k - 1}{\hat{t}_{k+1}}(\boldsymbol{x}_k - \boldsymbol{x}_{k-1})$
    $\hat{\boldsymbol{x}}_{k+1} := \mathrm{prox}_{\frac{1}{\hat{L}_{k+1}}\Psi}\left(\hat{\boldsymbol{y}}_{k+1} - \frac{1}{\hat{L}_{k+1}}\nabla f(\hat{\boldsymbol{y}}_{k+1})\right)$
    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
  $L_{k+1} = \hat{L}_{k+1},\ t_{k+1} = \hat{t}_{k+1}$
  $\boldsymbol{x}_{k+1} = \hat{\boldsymbol{x}}_{k+1}$
end for

and strong convexity parameter $\mu_{f'} = \mu$ whereas from (3.69) it follows that function Ψ' does not have known strong convexity. Therefore, Ψ' has the same properties as Ψ in Section 3.3 and f' is only slightly altered, which allows us to reuse many of the results obtained there. Naturally, the transfer in (3.68) and (3.69) does not alter the objective function, that is

$$F(\boldsymbol{x}) = f(\boldsymbol{x}) + \Psi(\boldsymbol{x}) = f'(\boldsymbol{x}) + \Psi'(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^n, \tag{3.70}$$

and gives rise to the following remarkable property.

Lemma 4. *The gap between the smooth part and its parabolic upper bound does not change with strong convexity transfer, namely*

$$Q_{f',L+\mu_\Psi,\boldsymbol{y}}(\boldsymbol{x}) - f(\boldsymbol{x}) = Q_{f,L,\boldsymbol{y}}(\boldsymbol{x}) - f'(\boldsymbol{x}), \quad \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n, \quad L > 0.$$

Proof. By expanding, rearranging, and canceling out terms we get

$$\begin{aligned} Q_{f',L+\mu_\Psi,\boldsymbol{y}} &= f'(\boldsymbol{y}) + \langle \nabla f'(\boldsymbol{y}), \boldsymbol{x} - \boldsymbol{y} \rangle + \frac{L'_{k+1}}{2}\|\boldsymbol{x} - \boldsymbol{y}\|_2^2 \\ &= f(\boldsymbol{y}) + \frac{\mu_\Psi}{2}\|\boldsymbol{y} - \boldsymbol{x}_0\|_2^2 + \langle \nabla f(\boldsymbol{y}) + \mu_\Psi(\boldsymbol{y} - \boldsymbol{x}_0), \boldsymbol{x} - \boldsymbol{y} \rangle \\ &\quad + \frac{L_{k+1} + \mu_\Psi}{2}\|\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \frac{\mu_\Psi}{2}\|\boldsymbol{x} - \boldsymbol{x}_0\|_2^2 - \frac{\mu_\Psi}{2}\|\boldsymbol{x} - \boldsymbol{x}_0\|_2^2 \\ &= f(\boldsymbol{y}) + \langle \nabla f(\boldsymbol{y}), \boldsymbol{x} - \boldsymbol{y} \rangle + \frac{L_{k+1}}{2}\|\boldsymbol{x} - \boldsymbol{y}\|_2^2 - \frac{\mu_\Psi}{2}\|\boldsymbol{x} - \boldsymbol{x}_0\|_2^2 \\ &= Q_{f,L,\boldsymbol{y}}(\boldsymbol{x}) + \frac{\mu_\Psi}{2}\|\boldsymbol{x} - \boldsymbol{x_0}\|_2^2, \quad \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n, \quad L > 0. \end{aligned} \tag{3.71}$$

Subtracting (3.68) from (3.71) completes the proof. □

Let $L'_{k+1} \overset{\text{def}}{=} L_{k+1} + \mu_\Psi$ for all $k \geq 0$. From Lemma 4 it follows that the descent condition for f', given by

$$f'(\boldsymbol{x}_{k+1}) \leq Q_{f',L'_{k+1},\boldsymbol{y}_{k+1}}(\boldsymbol{x}_{k+1}), \quad k \geq 0, \tag{3.72}$$

is *equivalent* at every iteration $k \geq 0$ to the corresponding condition for f, stated in (3.15).

By adding the equality in Lemma 4 and the objective invariance expression in (3.70), we obtain that the corresponding upper bounds are the same as well, namely

$$u_{k+1}(\boldsymbol{x}) = Q_{f',L'_{k+1},\boldsymbol{y}_{k+1}} + \Psi'(\boldsymbol{x}) = Q_{f,L_{k+1},\boldsymbol{y}_{k+1}} + \Psi(\boldsymbol{x}), \tag{3.73}$$

for all $\boldsymbol{x} \in \mathbb{R}^n$ and $k \geq 0$. Therefore, strong convexity transfer does not change the iterate update, that is

$$x_{k+1} = T_{f',\Psi',L'_{k+1}}(\boldsymbol{y}_{k+1}) = T_{f,\Psi,L_{k+1}}(\boldsymbol{y}_{k+1}), \quad k \geq 0. \tag{3.74}$$

3.4.2 Lower Bounds

In the strongly convex case, lower bounds can be constructed as a simple generalization of the relaxed supporting hyperplanes in (3.29).

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

$$\begin{aligned} \mathcal{R}_{L,\boldsymbol{y}}(\boldsymbol{x}) &\overset{\text{def}}{=} F(T_{f,\Psi,L}(\boldsymbol{y})) + \frac{1}{2L}\|g_L(\boldsymbol{y})\|_2^2 \\ &+ \langle g_L(\boldsymbol{y}), \boldsymbol{x} - \boldsymbol{y}\rangle + \frac{\mu}{2}\|\boldsymbol{x} - \boldsymbol{y}\|_2^2, \quad \boldsymbol{x} \in \mathbb{R}^n. \end{aligned} \tag{3.75}$$

The relaxed supporting generalized parabola is invariant to strong convexity transfer and, like the composite gradient, it need not be parametrized by f and Ψ.

Proposition 3. *If the descent condition in* (3.15) *holds at iteration $k \geq 0$, then the objective F is lower bounded as*

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

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

$$f'(\boldsymbol{x}) \geq f'(\boldsymbol{y}_{k+1}) + \left\langle \bigtriangledown f'(\boldsymbol{y}_{k+1}), \boldsymbol{x} - \boldsymbol{y}_{k+1}\right\rangle + \frac{\mu}{2}\|\boldsymbol{x} - \boldsymbol{y}_{k+1}\|_2^2, \tag{3.76}$$

for all $\boldsymbol{x} \in \mathbb{R}^n$ and $k \geq 0$. The definition of the composite gradient implies the existence of a vector $\boldsymbol{\xi}'_{k+1} \in \delta\Psi'(\boldsymbol{x}_{k+1})$ such that

$$g_{L'_{k+1}}(\boldsymbol{y}_{k+1}) = \bigtriangledown f'(\boldsymbol{y}_{k+1}) + \boldsymbol{\xi}'_{k+1}, \quad k \geq 0. \tag{3.77}$$

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

$$\begin{aligned}\Psi'(\boldsymbol{x}) &\geq \Psi'(\boldsymbol{x}_{k+1}) + \left\langle \boldsymbol{\xi}'_{k+1}, \boldsymbol{x} - \boldsymbol{x}_{k+1} \right\rangle \\ &\overset{(3.77)}{=} \Psi'(\boldsymbol{x}_{k+1}) + \left\langle g_{L'_{k+1}}(\boldsymbol{y}_{k+1}) - \bigtriangledown f'(\boldsymbol{y}_{k+1}), \boldsymbol{x} - \boldsymbol{x}_{k+1} \right\rangle, \end{aligned} \tag{3.78}$$

for all $\boldsymbol{x} \in \mathbb{R}^n$ and $k \geq 0$. Lemma 4 implies that the line search residual for f is the same as the one for f'. Therefore

$$\mathcal{N}_{k+1} = Q_{f, L_{k+1}, \boldsymbol{y}_{k+1}}(\boldsymbol{x}_{k+1}) - f(\boldsymbol{x}_{k+1}) = Q_{f', L'_{k+1}, \boldsymbol{y}_{k+1}}(\boldsymbol{x}_{k+1}) - f'(\boldsymbol{x}_{k+1}), \tag{3.79}$$

for all $k \geq 0$. The line search condition in (3.15) ensures that $\mathcal{N}_{k+1}$ is non-negative and negligibly small. We add together (3.76), (3.78), and the residual $\mathcal{N}_{k+1}$ in (3.79), rearrange terms along the lines of Subsection 3.3.1, and we reach the desired result. □

3.4.3 Generalizing ACGM to Arbitrary Strong Convexity

We construct ACGM in the strongly convex case using the same procedure as outlined in Subsection 3.3.2. The only difference lies in the choice of lower bounds. This time, the building blocks used in basic pattern Algorithm 2 are:

1. The composite parabolic upper bounds in (3.8).

2. The relaxed supporting parabola lower bounds from Proposition 3, given by

$$\begin{aligned} w_{k+1}(\boldsymbol{x}) = F(\boldsymbol{x}_{k+1}) + \frac{1}{2L'_{k+1}} \|g_{L'_{k+1}}(\boldsymbol{y}_{k+1})\|_2^2 \\ + \left\langle g_{L'_{k+1}}(\boldsymbol{y}_{k+1}), \boldsymbol{x} - \boldsymbol{y}_{k+1} \right\rangle + \frac{\mu}{2} \|\boldsymbol{x} - \boldsymbol{y}_{k+1}\|_2^2, \quad \boldsymbol{x} \in \mathbb{R}^n, \quad k \geq 0. \end{aligned} \tag{3.80}$$

3. The Lyapunov property of the gap sequence in (2.57) .

Using the same upper bounds as in Subsection 3.3.2 means that line 5 in Algorithm 2 is the proximal gradient step (3.12) and line 6 in Algorithm 2 is also (3.15) (see Subsection 3.4.1) .

The lower bounds are now generalized parabolae. Hence, the reasoning in Subsection 2.6.1 applies to ψ_k and ψ'_k, which are given by (2.46) and (2.47), respectively, for all $k \geq 0$. Substituting lower bound (3.80) in the estimate sequence update in line 15 of Algorithm 2 and differentiating with respect to $\boldsymbol{x}$ gives

$$\begin{aligned} \gamma_{k+1}(\boldsymbol{x} - \boldsymbol{v}_{k+1}) = \gamma_k(\boldsymbol{x} - \boldsymbol{v}_k) \\ + a_{k+1} \left(g_{L'_{k+1}}(\boldsymbol{y}_{k+1}) + \mu(\boldsymbol{x} - \boldsymbol{y}_{k+1}) \right), \quad \boldsymbol{x} \in \mathbb{R}^n, \quad k \geq 0, \end{aligned} \tag{3.81}$$

which leads, by matching the coefficients of the polynomials on both sides of (3.81), to the following curvature and vertex update rules for all $k \geq 0$:

$$\gamma_{k+1} = \gamma_k + a_{k+1}\mu, \tag{3.82}$$

$$\boldsymbol{v}_{k+1} = \frac{1}{\gamma_{k+1}}\left(\gamma_k \boldsymbol{v}_k - a_{k+1}(g_{L'_{k+1}}(\boldsymbol{y}_{k+1}) - \mu \boldsymbol{y}_{k+1})\right). \tag{3.83}$$

The update in (3.82) implies by induction that the curvature can be obtained directly from the convergence guarantees as

$$\gamma_k = \gamma_0 + \left(\sum_{i=1}^{k} a_i\right)\mu = (\gamma_0 - A_0\mu) + A_k\mu, \quad k \geq 0. \tag{3.84}$$

We proceed with the design of ACGM in the same way as in the non-strongly convex case. We formulate update rules for a_{k+1} and $\boldsymbol{y}_{k+1}$ in ACGM to ensure that (2.57) holds at every iteration $k \geq 0$ for any algorithmic state. Theorem 3 generalizes to arbitrary strong convexity as follows.

Theorem 4. *If at iteration $k \geq 0$, the descent condition for f in* (3.15) *holds, then*

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

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

$$\mathcal{A}_{k+1} \stackrel{\text{def}}{=} \frac{1}{2}\left(\frac{A_{k+1}}{L'_{k+1}} - \frac{a_{k+1}^2}{\gamma_{k+1}}\right)\|g_{L'_{k+1}}(\boldsymbol{y}_{k+1})\|_2^2, \tag{3.85}$$

$$\mathcal{B}_{k+1} \stackrel{\text{def}}{=} \frac{1}{\gamma_{k+1}}\left\langle g_{L'_{k+1}}(\boldsymbol{y}_{k+1}) - \frac{\mu}{2(A_k\gamma_{k+1} + a_{k+1}\gamma_k)}\boldsymbol{s}_{k+1}, \boldsymbol{s}_{k+1}\right\rangle, \tag{3.86}$$

$$\boldsymbol{s}_{k+1} \stackrel{\text{def}}{=} A_k\gamma_{k+1}\boldsymbol{x}_k + a_{k+1}\gamma_k\boldsymbol{v}_k - (A_k\gamma_{k+1} + a_{k+1}\gamma_k)\boldsymbol{y}_{k+1}. \tag{3.87}$$

Proof. As in the case of Theorem 3, we assume that $k \geq 0$ throughout the scope of this proof. When dealing with arbitrary strong convexity, the residual describing the tightness of the lower bound $w_{k+1}(\boldsymbol{x})$ at $\boldsymbol{x}$ is similarly given by

$$\begin{aligned} R_{k+1}(\boldsymbol{x}) \stackrel{\text{def}}{=}\ & F(\boldsymbol{x}) - F(\boldsymbol{x}_{k+1}) - \frac{1}{2L'_{k+1}}\|g_{L'_{k+1}}(\boldsymbol{y}_{k+1})\|_2^2 \\ & - \left\langle g_{L'_{k+1}}(\boldsymbol{y}_{k+1}), \boldsymbol{x} - \boldsymbol{y}_{k+1}\right\rangle - \frac{\mu}{2}\|\boldsymbol{x} - \boldsymbol{y}_{k+1}\|_2^2, \quad \boldsymbol{x} \in \mathbb{R}^n. \end{aligned} \tag{3.88}$$

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

$$\boldsymbol{G}_{k+1} \stackrel{\text{def}}{=} g_{L'_{k+1}}(\boldsymbol{y}_{k+1}) - \mu\boldsymbol{y}_{k+1}. \tag{3.89}$$

This simplifies the non-constant polynomial term in residual expression (3.88) as

$$\begin{aligned} & \left\langle g_{L'_{k+1}}(\boldsymbol{y}_{k+1}), \boldsymbol{x} - \boldsymbol{y}_{k+1}\right\rangle + \frac{\mu}{2}\|\boldsymbol{x} - \boldsymbol{y}_{k+1}\|_2^2 \\ = {} & \langle \boldsymbol{G}_{k+1} + \mu\boldsymbol{y}_{k+1}, \boldsymbol{x} - \boldsymbol{y}_{k+1}\rangle + \frac{\mu}{2}\|\boldsymbol{x}\|_2^2 + \frac{\mu}{2}\|\boldsymbol{y}_{k+1}\|_2^2 - \mu\langle \boldsymbol{x}, \boldsymbol{y}_{k+1}\rangle \\ = {} & \langle \boldsymbol{G}_{k+1}, \boldsymbol{x} - \boldsymbol{y}_{k+1}\rangle + \frac{\mu}{2}\|\boldsymbol{x}\|_2^2 - \frac{\mu}{2}\|\boldsymbol{y}_{k+1}\|_2^2. \end{aligned} \tag{3.90}$$

Proposition 3 ensures that $R_{k+1}(\boldsymbol{x}) \geq 0$ for all $\boldsymbol{x} \in \mathbb{R}^n$. As in Theorem 3, the central argument of the proof is built from the residual as

$$A_k R_{k+1}(\boldsymbol{x}_k) + a_{k+1} R_{k+1}(\boldsymbol{x}^*) \geq 0. \tag{3.91}$$

Note that while (3.91) has the same form as (3.35), (3.91) is more general.

By expanding the residual expressions using (3.88) and (3.90), (3.91) becomes

$$A_k(F(\boldsymbol{x}_k) - F(\boldsymbol{x}^*)) - A_{k+1}(F(\boldsymbol{x}_{k+1}) - F(\boldsymbol{x}^*)) \geq \mathcal{C}_{k+1}, \tag{3.92}$$

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

$$\begin{aligned} \mathcal{C}_{k+1} \overset{\text{def}}{=}\ & \mathcal{C}_{k+1}^{(1)} + \langle \boldsymbol{G}_{k+1}, A_k \boldsymbol{x}_k + a_{k+1}\boldsymbol{x}^* - A_{k+1}\boldsymbol{y}_{k+1}\rangle \\ & + \frac{A_k\mu}{2}\|\boldsymbol{x}_k\|_2^2 + \frac{a_{k+1}\mu}{2}\|\boldsymbol{x}^*\|_2^2 - \frac{A_{k+1}\mu}{2}\|\boldsymbol{y}_{k+1}\|_2^2, \end{aligned} \tag{3.93}$$

where

$$\mathcal{C}_{k+1}^{(1)} \overset{\text{def}}{=} \frac{A_{k+1}}{2L'_{k+1}}\|g_{L'_{k+1}}(\boldsymbol{y}_{k+1})\|_2^2. \tag{3.94}$$

Using the reduced composite gradient definition (3.89), we expand $\mathcal{C}_{k+1}^{(1)}$ as

$$\begin{aligned} \mathcal{C}_{k+1}^{(1)} &= \mathcal{A}_{k+1} + \frac{a_{k+1}^2}{2\gamma_{k+1}}\|\boldsymbol{G}_{k+1} + \mu\boldsymbol{y}_{k+1}\|_2^2 \\ &= \mathcal{A}_{k+1} + \mathcal{C}_{k+1}^{(2)} + \frac{a_{k+1}^2\mu}{\gamma_{k+1}}\langle \boldsymbol{G}_{k+1}, \boldsymbol{y}_{k+1}\rangle + \frac{a_{k+1}^2\mu^2}{2\gamma_{k+1}}\|\boldsymbol{y}_{k+1}\|_2^2, \end{aligned} \tag{3.95}$$

where

$$\mathcal{C}_{k+1}^{(2)} \overset{\text{def}}{=} \frac{a_{k+1}^2}{2\gamma_{k+1}}\|\boldsymbol{G}_{k+1}\|_2^2. \tag{3.96}$$

Combining the vertex update in (3.83) with (3.89), we obtain that

$$a_{k+1}\boldsymbol{G}_{k+1} = \gamma_k \boldsymbol{v}_k - \gamma_{k+1}\boldsymbol{v}_{k+1}. \tag{3.97}$$

Substituting (3.97) in $\mathcal{C}_{k+1}^{(2)}$ yields

$$\begin{aligned} \mathcal{C}_{k+1}^{(2)} &= \frac{1}{2\gamma_{k+1}}\|\gamma_k\boldsymbol{v}_k - \gamma_{k+1}\boldsymbol{v}_{k+1}\|_2^2 = \frac{\gamma_k^2}{2\gamma_{k+1}}\|\boldsymbol{v}_k\|_2^2 + \frac{\gamma_{k+1}}{2}\|\boldsymbol{v}_{k+1}\|_2^2 - \gamma_k\langle \boldsymbol{v}_k, \boldsymbol{v}_{k+1}\rangle \\ &= \frac{\gamma_{k+1}}{2}\|\boldsymbol{v}_{k+1}\|_2^2 - \frac{\gamma_k^2}{2\gamma_{k+1}}\|\boldsymbol{v}_k\|_2^2 + \frac{\gamma_k}{\gamma_{k+1}}\langle \gamma_k\boldsymbol{v}_k - \gamma_{k+1}\boldsymbol{v}_{k+1}, \boldsymbol{v}_k\rangle \\ &\overset{(3.97)}{=} \frac{\gamma_{k+1}}{2}\|\boldsymbol{v}_{k+1}\|_2^2 - \frac{\gamma_k}{2}\|\boldsymbol{v}_k\|_2^2 + \frac{\gamma_k\gamma_{k+1} - \gamma_k^2}{2\gamma_{k+1}}\|\boldsymbol{v}_k\|_2^2 + \frac{a\gamma_k}{\gamma_{k+1}}\langle \boldsymbol{G}_{k+1}, \boldsymbol{v}_k\rangle \\ &\overset{(3.82)}{=} \frac{\gamma_{k+1}}{2}\|\boldsymbol{v}_{k+1}\|_2^2 - \frac{\gamma_k}{2}\|\boldsymbol{v}_k\|_2^2 + \frac{\mu}{2\gamma_{k+1}}a_{k+1}\gamma_k\|\boldsymbol{v}_k\|_2^2 + \frac{1}{\gamma_{k+1}}\langle \boldsymbol{G}_{k+1}, a_{k+1}\gamma_k\boldsymbol{v}_k\rangle. \end{aligned} \tag{3.98}$$

We also define coefficient Y_{k+1} as

$$\begin{aligned} Y_{k+1} &\overset{\text{def}}{=} A_{k+1}\gamma_{k+1} - a_{k+1}^2\mu \overset{(3.82)}{=} (A_k + a_{k+1})(\gamma_k + a_{k+1}\mu) - a_{k+1}^2\mu \\ &= A_k\gamma_{k+1} + a_{k+1}\gamma_k. \end{aligned} \tag{3.99}$$

Combining (3.95) and (3.98) in (3.93), rearranging terms, and applying (3.99) yields

$$\mathcal{C}_{k+1} = \mathcal{A}_{k+1} + V_{k+1} + \frac{1}{\gamma_{k+1}} \langle \boldsymbol{G}_{k+1}, \boldsymbol{s}_{k+1} \rangle + \frac{\mu}{2\gamma_{k+1}} S_{k+1}, \tag{3.100}$$

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

$$S_{k+1} \stackrel{\text{def}}{=} A_k \gamma_{k+1} \|\boldsymbol{x}_k\|_2^2 + a_{k+1}\gamma_k \|\boldsymbol{v}_k\|_2^2 - Y_{k+1}\|\boldsymbol{y}_{k+1}\|_2^2, \tag{3.101}$$

$$V_{k+1} \stackrel{\text{def}}{=} \frac{\gamma_{k+1}}{2}\|\boldsymbol{v}_{k+1}\|_2^2 - \frac{\gamma_k}{2}\|\boldsymbol{v}_k\|_2^2 + \langle \boldsymbol{G}_{k+1}, a_{k+1}\boldsymbol{x}^* \rangle + \frac{a_{k+1}\mu}{2}\|\boldsymbol{x}^*\|_2^2. \tag{3.102}$$

Applying (3.97) and (3.82) in (3.102), we obtain

$$\begin{aligned} V_{k+1} &= \frac{\gamma_{k+1}}{2}\|\boldsymbol{v}_{k+1}\|_2^2 - \frac{\gamma_k}{2}\|\boldsymbol{v}_k\|_2^2 + \langle \gamma_k \boldsymbol{v}_k - \gamma_{k+1}\boldsymbol{v}_{k+1}, \boldsymbol{x}^* \rangle + \frac{\gamma_{k+1} - \gamma_k}{2}\|\boldsymbol{x}^*\|_2^2 \\ &= \frac{\gamma_{k+1}}{2}\|\boldsymbol{v}_{k+1} - \boldsymbol{x}^*\|_2^2 - \frac{\gamma_k}{2}\|\boldsymbol{v}_k - \boldsymbol{x}^*\|_2^2. \end{aligned} \tag{3.103}$$

Therefore, (3.102) is merely the straightforward generalization of (3.38) that handles arbitrary strong convexity.

Putting together (3.93), (3.100), and (3.103), we obtain

$$\Delta_{k+1} + \mathcal{A}_{k+1} + \frac{1}{\gamma_{k+1}} \langle \boldsymbol{G}_{k+1}, \boldsymbol{s}_{k+1} \rangle + \frac{\mu}{2\gamma_{k+1}} S_{k+1} \leq \Delta_k. \tag{3.104}$$

For brevity, we define ω_{k+1} as

$$\omega_{k+1} \stackrel{\text{def}}{=} \frac{a_{k+1}\gamma_k}{Y_{k+1}}. \tag{3.105}$$

This quantity will be used later on in formulating a monotone variant of ACGM. Residuals $\boldsymbol{s}_{k+1}$ and S_{k+1} can thus be written as

$$\boldsymbol{s}_{k+1} = Y_{k+1}((1-\omega_{k+1})\boldsymbol{x}_k + \omega_{k+1}\boldsymbol{v}_k - \boldsymbol{y}_{k+1}), \tag{3.106}$$

$$S_{k+1} = Y_{k+1}\left((1-\omega_{k+1})\|\boldsymbol{x}_k\|_2^2 + \omega_{k+1}\|\boldsymbol{v}_k\|_2^2 - \|\boldsymbol{y}_{k+1}\|_2^2\right). \tag{3.107}$$

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

$$\begin{aligned} (1-\omega_{k+1})\|\boldsymbol{x}_k\|_2^2 + \omega_{k+1}\|\boldsymbol{v}_k\|_2^2 = ((1-\omega_{k+1})\boldsymbol{x}_k + \omega_{k+1}\boldsymbol{v}_k)^2 \\ +(1-\omega_{k+1})\omega_{k+1}\|\boldsymbol{x}_k - \boldsymbol{v}_k\|_2^2. \end{aligned} \tag{3.108}$$

The proof of (3.108) is obtained simply by rearranging terms. Using (3.106) and (3.108) in (3.107) we obtain that

$$\begin{aligned} S_{k+1} &= Y_{k+1}\left(((1-\omega_{k+1})\boldsymbol{x}_k + \omega_{k+1}\boldsymbol{v}_k)^2 - \|\boldsymbol{y}_{k+1}\|_2^2\right) + S_{k+1}^{(1)} \\ &= \left\langle \frac{1}{Y_{k+1}}\boldsymbol{s}_{k+1} + 2\boldsymbol{y}_{k+1}, \boldsymbol{s}_{k+1} \right\rangle + S_{k+1}^{(1)}, \end{aligned} \tag{3.109}$$

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

$$\begin{aligned} S_{k+1}^{(1)} &\stackrel{\text{def}}{=} Y_{k+1}(1-\omega_{k+1})\omega_{k+1}\|\boldsymbol{x}_k - \boldsymbol{v}_k\|_2^2 \\ &= \frac{a_{k+1}A_k\gamma_k\gamma_{k+1}}{A_k\gamma_{k+1} + a_{k+1}\gamma_k}\|\boldsymbol{x}_k - \boldsymbol{v}_k\|_2^2. \end{aligned} \tag{3.110}$$

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

$$S_{k+1}^{(1)} \geq 0. \tag{3.111}$$

Putting together (3.89), (3.109), and (3.111), we obtain that

$$\begin{aligned}
&\frac{1}{\gamma_{k+1}} \langle \boldsymbol{G}_{k+1}, \boldsymbol{s}_{k+1} \rangle + \frac{\mu}{2\gamma_{k+1}} S_{k+1} \\
&\geq \frac{1}{\gamma_{k+1}} \left\langle \boldsymbol{G}_{k+1} + \frac{\mu}{2} \left(\frac{1}{Y_{k+1}} \boldsymbol{s}_{k+1} + 2\boldsymbol{y}_{k+1} \right), \boldsymbol{s}_{k+1} \right\rangle \\
&= \frac{1}{\gamma_{k+1}} \left\langle \boldsymbol{G}_{k+1} + \mu \boldsymbol{y}_{k+1} + \frac{\mu}{2Y_{k+1}} \boldsymbol{s}_{k+1}, \boldsymbol{s}_{k+1} \right\rangle \\
&= \frac{1}{\gamma_{k+1}} \left\langle g_{L'_{k+1}}(\boldsymbol{y}_{k+1}) + \frac{\mu}{2Y_{k+1}} \boldsymbol{s}_{k+1}, \boldsymbol{s}_{k+1} \right\rangle.
\end{aligned} \tag{3.112}$$

Combining (3.104) with (3.112) gives the desired result. □

Theorem 4 implies that the Lyapunov property of the gap sequence in (2.57) holds if, for any algorithmic state, $\mathcal{A}_{k+1} \geq 0$ and $\mathcal{B}_{k+1} \geq 0$. Again, because the vectors in inner product $\mathcal{B}_{k+1}$ may form an obtuse angle, we enforce $\mathcal{B}_{k+1} = 0$ by setting $\boldsymbol{s}_{k+1} = \mathbf{0}$, which leads to an update rule for the auxiliary point in the form of

$$\boldsymbol{y}_{k+1} = \frac{1}{A_k \gamma_{k+1} + a_{k+1}\gamma_k} \left(A_k \gamma_{k+1} \boldsymbol{x}_k + a_{k+1} \gamma_k \boldsymbol{v}_k \right), \quad k \geq 0. \tag{3.113}$$

Inequality $\mathcal{A}_{k+1} \geq 0$, by virtue of the non-negativity of $\|g_{L'_{k+1}}(\boldsymbol{y}_{k+1})\|_2^2$, becomes

$$A_{k+1}\gamma_{k+1} \geq a_{k+1}^2 L'_{k+1}, \quad k \geq 0. \tag{3.114}$$

Now that we have derived the expressions for the auxiliary point in (3.113) and the weights in (3.119), we can revert to original constituents f and Ψ and the corresponding LCE L_{k+1}. The accumulated weight update becomes

$$A_{k+1}\gamma_{k+1} \geq a_{k+1}^2 (L_{k+1} + \mu_\Psi), \quad k \geq 0, \tag{3.115}$$

which, using (3.5) and (3.82), can be cast as a quadratic inequality in a_{k+1}, given by

$$(L_{k+1} - \mu_f) a_{k+1}^2 - (\gamma_k + A_k \mu) a_{k+1} - A_k \gamma_k \leq 0, \quad k \geq 0. \tag{3.116}$$

Lemma 5. *At every iteration $k \geq 0$, we have that $L_{k+1} > \mu_f$.*

Proof. We have seen in Subsection 3.4.1 that transferring strong convexity between the two components f and Ψ does not alter the algorithm. This transfer can occur in the opposite direction and function $f(\boldsymbol{x}) - \frac{\mu_f}{2}\|\boldsymbol{x} - \boldsymbol{x}_0\|_2^2$ has the LCE given by $L_{k+1} - \mu_f$. Therefore $L_{k+1} - \mu_f > 0$. □

From (3.115) and Lemma 5, taking into account that $a_{k+1} > 0$, $A_k \geq 0$, $\gamma_k > 0$ for all $k \geq 0$, we obtain that

$$a_{k+1} \leq \mathcal{E}(\gamma_k, A_k, L_{k+1}), \quad k \geq 0, \tag{3.117}$$

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

$$\mathcal{E}(\gamma_k, A_k, L_{k+1}) \stackrel{\text{def}}{=} \frac{\gamma_k + A_k\mu + \sqrt{(\gamma_k + A_k\mu)^2 + 4(L_{k+1} - \mu_f)A_k\gamma_k}}{2(L_{k+1} - \mu_f)}. \tag{3.118}$$

We choose the most aggressive accumulated weight update by enforcing equality in (3.114). Thus, we obtain

$$(L_{k+1} + \mu_\Psi)a_{k+1}^2 = A_{k+1}\gamma_{k+1}, \tag{3.119}$$

which translates into an update rule for a_{k+1}, given by

$$a_{k+1} = \mathcal{E}(\gamma_k, A_k, L_{k+1}). \tag{3.120}$$

The vertex update in (3.83) can be written without the composite gradient for all $k \geq 0$ as

$$\begin{aligned}\gamma_{k+1}\boldsymbol{v}_{k+1} &= \gamma_k\boldsymbol{v}_k - a_{k+1}(L'_{k+1}(\boldsymbol{y}_{k+1} - \boldsymbol{x}_{k+1}) - \mu\boldsymbol{y}_{k+1}) \\ &= \gamma_k\boldsymbol{v}_k + a_{k+1}(L_{k+1} + \mu_\Psi)\boldsymbol{x}_{k+1} - a_{k+1}(L_{k+1} - \mu_f)\boldsymbol{y}_{k+1}. \end{aligned} \tag{3.121}$$

By inserting in Algorithm 2 the process $\mathcal{C}_{a,u,w}$ comprising the weight update in (3.120) and the iterate generator in (3.12), together with the LSSC in (3.15) and the vertex update in (3.121), we obtain a variant of ACGM for objectives with arbitrary strong convexity as listed in Algorithm 5. Note that scale invariance applies here as well and we can assume that $\gamma_0 = 1$. However, Nesterov's FGM does not make this assumption and for better comparison with this method, we impose no restrictions on γ_0 in ACGM at this stage, besides positivity.

Extrapolated form

We show that ACGM can be written in an extrapolated form for strongly-convex objectives as well. First, Lemma 3 generalizes without modification.

Lemma 6. *When dealing with arbitrary strong convexity, the estimate sequence vertices can be obtained from main iterates through extrapolation, namely*

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

Proof. This time, for the sake of brevity, we utilize only analytical argu-

Algorithm 5 ACGM in estimate sequence form

ACGM$(\boldsymbol{x}_0, L_0, A_0, \gamma_0, r_u, r_d, \mu_f, \mu_\Psi, K)$

$\boldsymbol{v}_0 = \boldsymbol{x}_0$
$\mu = \mu_f + \mu_\Psi$
for $k = 0, ..., K-1$ **do**
  $\hat{L}_{k+1} := r_d L_k$
  loop
    $\hat{a}_{k+1} := \frac{1}{2(\hat{L}_{k+1}-\mu_f)}\left(\gamma_k + A_k\mu + \sqrt{(\gamma_k + A_k\mu)^2 + 4(\hat{L}_{k+1}-\mu_f)A_k\gamma_k}\right)$
    $\hat{A}_{k+1} := A_k + \hat{a}_{k+1}$
    $\hat{\gamma}_{k+1} := \gamma_k + \hat{a}_{k+1}\mu$
    $\hat{\boldsymbol{y}}_{k+1} := \frac{1}{A_k\hat{\gamma}_{k+1} + \hat{a}_{k+1}\gamma_k}(A_k\hat{\gamma}_{k+1}\boldsymbol{x}_k + \hat{a}_{k+1}\gamma_k\boldsymbol{v}_k)$
    $\hat{\boldsymbol{x}}_{k+1} := \mathrm{prox}_{\frac{1}{\hat{L}_{k+1}}\Psi}\left(\hat{\boldsymbol{y}}_{k+1} - \frac{1}{\hat{L}_{k+1}}\nabla f(\hat{\boldsymbol{y}}_{k+1})\right)$
    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
  $L_{k+1} = \hat{L}_{k+1},\ A_{k+1} = \hat{A}_{k+1},\ a_{k+1} = \hat{a}_{k+1},\ \gamma_{k+1} = \hat{\gamma}_{k+1}$
  $\boldsymbol{x}_{k+1} = \hat{\boldsymbol{x}}_{k+1},\ \boldsymbol{y}_{k+1} = \hat{\boldsymbol{y}}_{k+1}$
  $\boldsymbol{v}_{k+1} = \frac{1}{\gamma_{k+1}}(\gamma_k\boldsymbol{v}_k + a_{k+1}(L_{k+1}+\mu_\Psi)\boldsymbol{x}_{k+1} - a_{k+1}(L_{k+1}-\mu_f)\boldsymbol{y}_{k+1})$
end for

ments and omit the visual proof. Combining (3.113) with (3.121) yields

$$\begin{aligned}\boldsymbol{v}_{k+1} &= \frac{\gamma_k}{\gamma_{k+1}}\frac{(a_{k+1}\gamma_k + A_k\gamma_{k+1})\boldsymbol{y}_{k+1} - A_k\gamma_{k+1}\boldsymbol{x}_k}{a_{k+1}\gamma_k} \\ &\quad + \frac{a_{k+1}(L_{k+1}+\mu_\Psi)}{\gamma_{k+1}}\boldsymbol{x}_{k+1} - \frac{a_{k+1}(L_{k+1}-\mu_f)}{\gamma_{k+1}}\boldsymbol{y}_{k+1} \\ &= \frac{a_{k+1}\gamma_k + A_k\gamma_{k+1} - a_{k+1}^2(L_{k+1}-\mu_f)}{a_{k+1}\gamma_{k+1}}\boldsymbol{y}_{k+1} \\ &\quad + \frac{a_{k+1}(L_{k+1}+\mu_\Psi)}{\gamma_{k+1}}\boldsymbol{x}_{k+1} - \frac{A_k}{a_{k+1}}\boldsymbol{x}_k, \quad k \ge 0. \end{aligned} \tag{3.122}$$

Two subexpressions in (3.122) can be simplified using (3.119). First, the coefficient of $\boldsymbol{y}_{k+1}$ is proportional to

$$\begin{aligned}&a_{k+1}\gamma_k + A_k\gamma_{k+1} - a_{k+1}^2(L_{k+1}-\mu_f) \\ &= a_{k+1}\gamma_k + A_k\gamma_{k+1} - A_{k+1}\gamma_{k+1} - a_{k+1}^2\mu = 0, \quad k \ge 0.\end{aligned} \tag{3.123}$$

Also, the coefficient of $\boldsymbol{x}_{k+1}$ can be written as

$$\frac{a_{k+1}(L_{k+1}+\mu_\Psi)}{\gamma_{k+1}} = \frac{A_{k+1}}{a_{k+1}}, \quad k \ge 0. \tag{3.124}$$

Finally, we eliminate the $\boldsymbol{y}_{k+1}$ term in (3.122) based on (3.123), we rewrite the coefficient of $\boldsymbol{x}_{k+1}$ using (3.124), and get the desired result. □

Sequence $\{t_k\}_{k\geq 0}$ generalizes here to

$$t_k \overset{\text{def}}{=} \sqrt{\frac{(L_k+\mu_\Psi)A_k}{\gamma_k}}, \quad k \geq 0. \tag{3.125}$$

The terms of this sequence are vertex extrapolation factors for $k \geq 1$ because

$$t_{k+1} = \sqrt{\frac{(L_{k+1}+\mu_\Psi)A_{k+1}}{\gamma_{k+1}}} \overset{(3.119)}{=} \sqrt{\frac{A_{k+1}^2}{a_{k+1}^2}} = \frac{A_{k+1}}{a_{k+1}}, \quad k \geq 0. \tag{3.126}$$

Based on Lemma 6, we seek a generalization of Proposition 1 to support arbitrary strong convexity. To use extrapolation in the first iteration we maintain the artificial iterate $\boldsymbol{x}_{-1} \overset{\text{def}}{=} \boldsymbol{x}_0$. We introduce the local inverse condition number

$$q_k \overset{\text{def}}{=} \frac{\mu}{L_k'} = \frac{\mu}{L_k+\mu_\Psi}, \quad k \geq 0. \tag{3.127}$$

Using the notation in (3.125) and (3.127), we can express the curvature ratio γ_k/γ_{k+1} as

$$\begin{aligned}\frac{\gamma_k}{\gamma_{k+1}} &= 1 - \frac{a_{k+1}\mu}{\gamma_{k+1}} = 1 - \frac{A_{k+1}a_{k+1}\mu}{A_{k+1}\gamma_{k+1}} \overset{(3.119)}{=} 1 - \frac{a_{k+1}A_{k+1}\mu}{(L_{k+1}+\mu_\Psi)a_{k+1}^2} \\ &= 1 - q_{k+1}t_{k+1}, \quad k \geq 0. \end{aligned} \tag{3.128}$$

Proposition 4. *At every iteration $k \geq 0$, the new auxiliary point y_{k+1} can be obtained from the previous iterates by extrapolation as*

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

where

$$\beta_k = \frac{t_k - 1}{t_{k+1}}\frac{1 - q_{k+1}t_{k+1}}{1 - q_{k+1}}.$$

Proof. For $k = 0$, (3.113) implies that

$$\begin{aligned}\boldsymbol{y}_1 &= \frac{1}{A_0\gamma_1 + a_1\gamma_0}\left(A_0\gamma_1\boldsymbol{x}_0 + a_1\gamma_0\boldsymbol{v}_0\right) = \boldsymbol{x}_0 \\ &= \boldsymbol{x}_0 + \frac{t_0 - 1}{t_1}\frac{1 - q_1t_1}{1 - q_1}(\boldsymbol{x}_0 - \boldsymbol{x}_{-1}). \end{aligned} \tag{3.129}$$

When $k \geq 1$, from (3.113) and Lemma 6 we have that

$$\begin{aligned}\boldsymbol{y}_{k+1} &= \frac{1}{A_k\gamma_{k+1} + a_{k+1}\gamma_k}\left(A_k\gamma_{k+1}\boldsymbol{x}_k + a_{k+1}\gamma_k(\boldsymbol{x}_{k-1} + t_k(\boldsymbol{x}_k - \boldsymbol{x}_{k-1}))\right) \\ &= \boldsymbol{x}_k + \frac{1}{A_k\gamma_{k+1} + a_{k+1}\gamma_k}\left(a_{k+1}\gamma_k(t_k - 1)\boldsymbol{x}_k + a_{k+1}\gamma_k(1 - t_k)\boldsymbol{x}_{k-1}\right) \\ &= \boldsymbol{x}_k + \beta_k(\boldsymbol{x}_k - \boldsymbol{x}_{k-1}), \end{aligned} \tag{3.130}$$

where

$$\beta_k = \frac{a_{k+1}\gamma_k\,(t_k-1)}{A_k\gamma_{k+1}+a_{k+1}\gamma_k} \overset{(3.126)}{=} \frac{t_k-1}{t_{k+1}}\frac{A_{k+1}\gamma_k}{A_k\gamma_{k+1}+a_{k+1}\gamma_k} \overset{(3.5)}{=} \frac{t_k-1}{t_{k+1}}\frac{\frac{\gamma_k}{\gamma_{k+1}}}{\frac{A_{k+1}\gamma_{k+1}-a_{k+1}(\gamma_{k+1}-\gamma_k)}{A_{k+1}\gamma_{k+1}}} \overset{(3.82)}{=} \frac{t_k-1}{t_{k+1}}\frac{\frac{\gamma_k}{\gamma_{k+1}}}{1-\frac{\mu a_{k+1}^2}{A_{k+1}\gamma_{k+1}}} \overset{(3.119)}{=} \frac{t_k-1}{t_{k+1}}\frac{\frac{\gamma_k}{\gamma_{k+1}}}{1-q_{k+1}} \overset{(3.128)}{=} \frac{t_k-1}{t_{k+1}}\frac{1-q_{k+1}t_{k+1}}{1-q_{k+1}}. \tag{3.131}$$

□

Definition (3.125) and ratio (3.128) facilitate the derivation of a recursion rule for t_k that does not depend on either a_k, A_k, or γ_k for all $k \geq 0$ and $\mu \geq 0$ as follows.

Proposition 5. *For arbitrary strong convexity, the vertex extrapolation factors obey the following recursion rule:*

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

Proof. Definition (3.125) can be reformulated as

$$(L_k+\mu_\Psi)A_k = \gamma_k t_k^2, \quad k \geq 0. \tag{3.132}$$

Combining (3.125) with (3.119) yields

$$(L_{k+1}+\mu_\Psi)a_{k+1} = \gamma_{k+1}t_{k+1}, \quad k \geq 0. \tag{3.133}$$

Given the positivity of LCEs, the weight recursion in (3.5) can be written for all $k \geq 0$ as

$$(L_{k+1}+\mu_\Psi)A_{k+1} - (L_{k+1}+\mu_\Psi)a_{k+1} - \frac{L_{k+1}+\mu_\Psi}{L_k+\mu_\Psi}(L_k+\mu_\Psi)A_k = 0. \tag{3.134}$$

Using (3.132) and (3.133), we can replace the convergence guarantees in (3.134) as

$$\gamma_{k+1}t_{k+1}^2 - \gamma_{k+1}t_{k+1} - \frac{L_{k+1}+\mu_\Psi}{L_k+\mu_\Psi}\gamma_k t_k^2 = 0, \quad k \geq 0. \tag{3.135}$$

Dividing by γ_{k+1} followed the substitution of the curvature ratio in (3.128) completes the proof. □

The quadratic equation in Proposition 5 has only one positive root. Its expression provides an update rule for t_{k+1} as

$$t_{k+1} = \frac{1}{2}\left(1-q_k t_k^2 + \sqrt{\left(1-q_k t_k^2\right)^2 + 4\frac{L_{k+1}+\mu_\Psi}{L_k+\mu_\Psi}t_k^2}\right), \quad k \geq 0. \tag{3.136}$$

Now, from Proposition 4 and (3.136) we can formulate ACGM for arbitrary strong convexity using iterate extrapolation, as presented in Algorithm 6. Algorithms 5 and 6 perform calculations differently, but are guaranteed to generate the same iterates. Also, Algorithms 3 and 4 are particular cases ($\mu = \mu_f = \mu_\Psi = 0$) of Algorithms 5 and 6, respectively.

Algorithm 6 ACGM in extrapolated form
ACGM$(\boldsymbol{x}_0, L_0, A_0, \gamma_0, r_u, r_d, \mu_f, \mu_\Psi, K)$

$\boldsymbol{x}_{-1} = \boldsymbol{x}_0$
$\mu = \mu_f + \mu_\Psi$
$q_0 = \frac{\mu}{L_0+\mu_\Psi}$
$t_0 = \sqrt{\frac{(L_0+\mu_\Psi)A_0}{\gamma_0}}$
for $k = 0, ..., K-1$ **do**
  $\hat{L}_{k+1} := r_d L_k$
  loop
    $\hat{q}_{k+1} := \frac{\mu}{\hat{L}_{k+1}+\mu_\Psi}$
    $\hat{t}_{k+1} := \frac{1}{2}\left(1 - q_k t_k^2 + \sqrt{(1-q_k t_k^2)^2 + 4\frac{\hat{L}_{k+1}+\mu_\Psi}{L_k+\mu_\Psi} t_k^2}\right)$
    $\hat{\boldsymbol{y}}_{k+1} := \boldsymbol{x}_k + \frac{t_k - 1}{\hat{t}_{k+1}} \frac{1-\hat{q}_{k+1}\hat{t}_{k+1}}{1-\hat{q}_{k+1}} (\boldsymbol{x}_k - \boldsymbol{x}_{k-1})$
    $\hat{\boldsymbol{x}}_{k+1} := \mathbf{prox}_{\frac{1}{\hat{L}_{k+1}}\Psi}\left(\hat{\boldsymbol{y}}_{k+1} - \frac{1}{\hat{L}_{k+1}}\nabla f(\hat{\boldsymbol{y}}_{k+1})\right)$
    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
  $L_{k+1} = \hat{L}_{k+1},\ q_{k+1} = \hat{q}_{k+1},\ t_{k+1} = \hat{t}_{k+1}$
  $\boldsymbol{x}_{k+1} = \hat{\boldsymbol{x}}_{k+1}$
end for

Retrieving the convergence guarantee

In Algorithm 5, the convergence guarantee in expression (2.5) is obtained directly from a single state variable A_k. For Algorithm 6, we need to retrieve the convergence guarantees from the other state parameters.

First, we determine how much information is contained in the vertex extrapolation factors by investigating their behavior. By putting together (3.84) and (3.125), we obtain a closed form expression for the vertex extrapolation factors as

$$t_k = \sqrt{\frac{(L_k+\mu_\Psi)A_k}{(\gamma_0 - A_0\mu) + A_k\mu}} = \sqrt{\frac{L_k+\mu_\Psi}{\frac{\gamma_0 - A_0\mu}{A_k} + \mu}}, \quad k \geq 0. \tag{3.137}$$

The following statements are direct consequences of (3.137).

If $\gamma_0 > A_0\mu$, then

$$t_k < \frac{1}{\sqrt{q_k}}, \quad k \geq 0. \tag{3.138}$$

If $\gamma_0 = A_0\mu$, then

$$t_k = \frac{1}{\sqrt{q_k}}, \quad k \geq 0. \tag{3.139}$$

If $\gamma_0 < A_0\mu$, then

$$t_k > \frac{1}{\sqrt{q_k}}, \quad k \geq 0. \tag{3.140}$$

Note that the parameter choices in (3.139) and (3.140) are valid in the strongly convex case.

From an asymptotic perspective, the convergence guarantees can be arbitrarily large and therefore, for any parameter setup, we have that

$$\lim_{k\to\infty} q_k t_k^2 = 1. \tag{3.141}$$

The difference in behavior of the vertex extrapolation factors leads us to distinguish between two scenarios. The most common one is outlined in the following lemma.

Lemma 7. *If $\gamma_0 \neq A_0\mu$, then*

$$A_k = \frac{(\gamma_0 - A_0\mu)t_k^2}{(L_k + \mu_\Psi)(1 - q_k t_k^2)}, \quad k \geq 1.$$

Proof. Definition (3.125) gives

$$(L_k + \mu_\Psi)A_k = \gamma_k t_k^2, \quad k \geq 0. \tag{3.142}$$

Substituting curvature γ_k given by (3.84) in (3.125), we obtain that

$$1 - q_k t_k^2 = 1 - \frac{\mu}{L_k + \mu_\Psi}\frac{(L_k + \mu_\Psi)A_k}{\gamma_k} = \frac{\gamma_k - \mu A_k}{\gamma_k} = \frac{\gamma_0 - A_0\mu}{\gamma_k}, \tag{3.143}$$

for all $k \geq 0$. Combining (3.142) with (3.143) yields

$$(L_k + \mu_\Psi)(1 - q_k t_k^2)A_k = (\gamma_0 - A_0\mu)t_k^2, \quad k \geq 0. \tag{3.144}$$

When $\gamma_0 \neq A_0\mu$, (3.138) and (3.140) imply that $1 - q_k t_k^2 \neq 0$. Dividing by this non-zero quantity in (3.144), we get the desired result. □

Consequently, if $\gamma_0 \neq A_0\mu$, the convergence guarantee can be derived directly from the state parameters without alterations to Algorithm 6.

Border-case

When $\gamma_0 = A_0\mu$, Algorithm 6 can be brought to a simpler form. From (3.139) we have that the auxiliary point extrapolation factor in Proposition 4 is given by

$$\begin{aligned}\beta_k &= \frac{\frac{1}{\sqrt{q_k}} - 1}{\frac{1}{\sqrt{q_{k+1}}}}\frac{1 - \frac{q_{k+1}}{\sqrt{q_{k+1}}}}{1 - q_{k+1}} = \frac{\sqrt{q_{k+1}}}{\sqrt{q_k}}\frac{1 - \sqrt{q_k}}{1 + \sqrt{q_{k+1}}} \\ &= \frac{\sqrt{L_k + \mu_\Psi} - \sqrt{\mu}}{\sqrt{L_{k+1} + \mu_\Psi} + \sqrt{\mu}}, \qquad k \geq 0.\end{aligned} \tag{3.145}$$

However, the sequence $\{t_k\}_{k\geq 0}$ does not store any relevant information and can be left out. This means that the convergence guarantee A_k requires a

dedicated update. Assumption $\gamma_0 = A_0\mu$ in (3.84) leads to $\gamma_k = \mu A_k$ for all $k \geq 0$. The curvature ratio in (3.128) becomes

$$\frac{\gamma_k}{\gamma_{k+1}} = \frac{\mu A_k}{\mu A_{k+1}} = 1 - q_{k+1}t_{k+1} = 1 - \sqrt{q_{k+1}}, \quad k \geq 0, \tag{3.146}$$

which provides a simple recursion rule in the form of

$$A_{k+1} = \frac{1}{1 - \sqrt{q_{k+1}}} A_k = \frac{\sqrt{L_{k+1} + \mu_\Psi}}{\sqrt{L_{k+1} + \mu_\Psi} - \sqrt{\mu}} A_k, \quad k \geq 0. \tag{3.147}$$

Due to scaling invariance, we can select any pair (A_0, γ_0) that is a positive multiple of $(1, \mu)$. For simplicity, we choose $A_0 = 1$ and $\gamma_0 = \mu$.

The local inverse condition number sequence $\{q_k\}_{k \geq 0}$ does not appear in updates (3.145) and (3.147). Hence, it can also be abstracted away. The form taken by ACGM in this border-case, after simplifications, is listed in Algorithm 7.

Algorithm 7 Border-case ACGM (BACGM) in extrapolated form

BACGM($\boldsymbol{x}_0, L_0, r_u, r_d, \mu_f, \mu_\Psi, K$)

$\boldsymbol{x}_{-1} = \boldsymbol{x}_0$
$\mu = \mu_f + \mu_\Psi$
$A_0 = 1$
for $k = 0, ..., K-1$ **do**
  $\hat{L}_{k+1} := r_d L_k$
  loop
    $\hat{\boldsymbol{y}}_{k+1} := \boldsymbol{x}_k + \frac{\sqrt{\hat{L}_{k+1} + \mu_\Psi} - \sqrt{\mu}}{\sqrt{\hat{L}_{k+1} + \mu_\Psi} + \sqrt{\mu}} (\boldsymbol{x}_k - \boldsymbol{x}_{k-1})$
    $\hat{\boldsymbol{x}}_{k+1} := \mathrm{prox}_{\frac{1}{\hat{L}_{k+1}}\Psi} \left(\hat{\boldsymbol{y}}_{k+1} - \frac{1}{\hat{L}_{k+1}} \nabla f(\hat{\boldsymbol{y}}_{k+1}) \right)$
    if $f(\hat{\boldsymbol{x}}_{k+1}) \leq Q_{f, \hat{L}_{k+1}, \hat{\boldsymbol{y}}_{k+1}}(\hat{\boldsymbol{z}})$ **then**
      Break from loop
    else
      $\hat{L}_{k+1} := r_u \hat{L}_{k+1}$
    end if
  end loop
  $L_{k+1} := \hat{L}_{k+1}$
  $\boldsymbol{x}_{k+1} := \hat{\boldsymbol{x}}_{k+1}$
  $A_{k+1} := \frac{\sqrt{L_{k+1} + \mu_\Psi}}{\sqrt{L_{k+1} + \mu_\Psi} - \sqrt{\mu}} A_k$
end for

3.5 Monotone ACGM

Even though the convergence guarantee upper bound in the ISDUB expression (2.5) is monotonically decreasing, the variants of ACGM derived up to this point cannot guarantee that the objective value at the current iterate $F(\boldsymbol{x}_k)$ is monotonically decreasing for all $k \geq 0$. Monotonicity is a desirable property, because it prevents divergence when dealing with proximal operators that lack a closed form expression and need to be approximated or other kinds of inexact oracles [6, 22]. Generally, monotonicity leads to a more stable and predictable convergence rate.

3.5.1 Upper Bounds

We want ACGM to converge as fast as possible while maintaining the monotonicity property, expressed as

$$F(\boldsymbol{x}_{k+1}) \leq F(\boldsymbol{x}_k), \quad k \geq 0. \tag{3.148}$$

Then, without further knowledge of the objective function, the simple upper bound in (3.8) and (3.148) suggest a simple expression of the monotone ACGM upper bound in the form of

$$u_{k+1}(\boldsymbol{x}) = \min\{Q_{f,L_{k+1},\boldsymbol{y}_{k+1}}(\boldsymbol{x}) + \Psi(\boldsymbol{x}), F(\boldsymbol{x}_k) + \sigma_{\{\boldsymbol{x}_k\}}(\boldsymbol{x})\}, \quad k \geq 0, \tag{3.149}$$

where σ_X is the indicator function [35] of set X, given by

$$\sigma_X(\boldsymbol{x}) = \begin{cases} 0, & \boldsymbol{x} \in X, \\ +\infty, & \text{otherwise.} \end{cases} \tag{3.150}$$

Note that the upper bound in (3.149) differs from the one described in Subsection 3.2.1 and therefore the new iterate $\boldsymbol{x}_{k+1}$ is no longer given by (3.12). Instead, it is generated by

$$\boldsymbol{x}_{k+1} = \arg\min\{F(\boldsymbol{z}_{k+1}), F(\boldsymbol{x}_k)\}, \quad k \geq 0, \tag{3.151}$$

where

$$\boldsymbol{z}_{k+1} = T_{f,\Psi,L_{k+1}}(\boldsymbol{y}_{k+1}), \quad k \geq 0. \tag{3.152}$$

Therefore, most properties of $\boldsymbol{x}_{k+1}$ in Section 3.4 apply only to $\boldsymbol{z}_{k+1}$ in this section, most notable being the descent condition, now given by

$$f(\boldsymbol{z}_{k+1}) \leq Q_{f,L_{k+1},\boldsymbol{y}_{k+1}}(\boldsymbol{z}_{k+1}), \quad k \geq 0. \tag{3.153}$$

3.5.2 Formulating Monotone ACGM

To construct an algorithm, we also need an expression for the lower bound w_{k+1}. Since computing $\boldsymbol{z}_{k+1}$ in (3.152) requires two oracle calls, we reuse

this information to construct a lower bound based on Proposition 3 as

$$w_{k+1}(\boldsymbol{x}) = \mathcal{R}_{L'_{k+1},\boldsymbol{y}_{k+1}}(\boldsymbol{x}) = F(\boldsymbol{z}_{k+1}) + \frac{1}{2L'_{k+1}}\|g_{L'_{k+1}}(\boldsymbol{y}_{k+1})\|_2^2$$
$$+ \left\langle g_{L'_{k+1}}(\boldsymbol{y}_{k+1}), \boldsymbol{x} - \boldsymbol{y}_{k+1} \right\rangle + \frac{\mu}{2}\|\boldsymbol{x} - \boldsymbol{y}_{k+1}\|_2^2, \quad \boldsymbol{x} \in \mathbb{R}^n, \quad k \geq 0. \tag{3.154}$$

As before, we construct monotone ACGM (MACGM) based on Algorithm 2, but using the new upper and lower bounds. Therefore, the building blocks in Algorithm 2 are as follows:

1. The upper bounds in (3.149).

2. The relaxed supporting parabola lower bounds from (3.154).

3. The Lyapunov property of the gap sequence in (2.57) .

From the the choice of upper bounds in (3.149), it follows that line 5 in Algorithm 2 is given by (3.151) and the LSSC (line 6 in Algorithm 2) is now (3.153).

The lower bounds in (3.154) are generalized parabolae. Consequently, the results in Subsection 2.6.1 hold for z_{k+1} and, following the procedure outlined in Subsection 3.4.3, we get the estimate function curvature and the weight update for all $k \geq 0$ as

$$\gamma_{k+1} = \gamma_k + a_{k+1}\mu, \tag{3.155}$$

$$\boldsymbol{v}_{k+1} = \frac{1}{\gamma_{k+1}}\left(\gamma_k \boldsymbol{v}_k + a_{k+1}(L_{k+1} + \mu_\Psi)\boldsymbol{z}_{k+1} - a_{k+1}(L_{k+1} - \mu_f)\boldsymbol{y}_{k+1}\right). \tag{3.156}$$

The descent condition in (3.153) can be equivalently expressed in terms of composite objective values (see also (3.14) and (3.15)) as

$$F(\boldsymbol{z}_{k+1}) \leq Q_{f,L_{k+1},\boldsymbol{y}_{k+1}}(\boldsymbol{z}_{k+1}) + \Psi(\boldsymbol{z}_{k+1}), \quad \boldsymbol{x} \in \mathbb{R}^n. \tag{3.157}$$

In Theorem 4 we have shown that if we choose $\boldsymbol{x}_{k+1} = \boldsymbol{z}_{k+1}$, we can build a method that maintains a monotone gap sequence. However, it is possible to choose a point $\boldsymbol{x}_{k+1}$ that is better than $\boldsymbol{z}_{k+1}$ using the following result.

Theorem 5. *If at iteration $k \geq 0$ we have*

$$F(\boldsymbol{x}_{k+1}) \leq F(\boldsymbol{z}_{k+1}) \leq Q_{f,L_{k+1},\boldsymbol{y}_{k+1}}(\boldsymbol{z}_{k+1}) + \Psi(\boldsymbol{z}_{k+1}),$$

then

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

where subexpressions $\mathcal{A}_{k+1}$ and $\mathcal{B}_{k+1}$ are given by (3.85) *and* (3.86)*, respectively, without modification.*

Proof. As in Theorems 3 and 4, we assume that $k \geq 0$ throughout this proof. We define residual $R_{k+1}(\boldsymbol{x})$ as

$$R_{k+1}(\boldsymbol{x}) \stackrel{\text{def}}{=} F(\boldsymbol{x}) - w_{k+1}(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^n, \tag{3.158}$$

where $w_{k+1}(\boldsymbol{x})$ is given by (3.154). The lower bound property of $w_{k+1}(\boldsymbol{x})$ means that $R_{k+1}(\boldsymbol{x}) \geq 0$ for all $\boldsymbol{x} \in \mathbb{R}^n$. Therefore

$$A_k R_{k+1}(\boldsymbol{x}_k) + a_{k+1} R_{k+1}(\boldsymbol{x}^*) \geq 0. \tag{3.159}$$

The terms in (3.159) can be expanded and rearranged to yield

$$A_k(F(\boldsymbol{x}_k) - F(\boldsymbol{x}^*)) - A_{k+1}(F(\boldsymbol{z}_{k+1}) - F(\boldsymbol{x}^*)) \geq \mathcal{C}_{k+1}, \tag{3.160}$$

where the lower bound $\mathcal{C}_{k+1}$ is given by (3.93). The form in (3.93) can be equivalently expressed as (3.100) and we have that (3.112) holds. The justification carries over as is from the proof of Theorem 4.

By combining (3.100) with (3.112), we obtain that

$$\mathcal{C}_{k+1} \geq \mathcal{A}_{k+1} + V_{k+1} + \frac{1}{\gamma_{k+1}} \left\langle g_{L'_{k+1}}(\boldsymbol{y}_{k+1}) + \frac{\mu}{2Y_{k+1}} \boldsymbol{s}_{k+1}, \boldsymbol{s}_{k+1} \right\rangle. \tag{3.161}$$

Putting together (3.160) and (3.161), rearranging terms, and applying $F(\boldsymbol{x}_{k+1}) \leq F(\boldsymbol{z}_{k+1})$ gives the desired result. □

Theorem 5 provides a simple sufficient condition for the monotonicity of the gap sequence, regardless of the algorithmic state, given for all $k \geq 0$ by the following relations:

$$\boldsymbol{y}_{k+1} = \frac{1}{A_k \gamma_{k+1} + a_{k+1}\gamma_k} \left(A_k \gamma_{k+1} \boldsymbol{x}_k + a_{k+1} \gamma_k \boldsymbol{v}_k \right), \tag{3.162}$$

$$(L_{k+1} + \mu_\Psi) a_{k+1}^2 \leq A_{k+1} \gamma_{k+1}. \tag{3.163}$$

The latter can be written as

$$a_{k+1} \leq \mathcal{E}(\gamma_k, A_k, L_{k+1}), \quad k \geq 0, \tag{3.164}$$

where expression $\mathcal{E}(\gamma_k, A_k, L_{k+1})$ is given by (3.118). To provide the best convergence guarantees, as before, we enforce equality in (3.117), namely

$$a_{k+1} = \mathcal{E}(\gamma_k, A_k, L_{k+1}). \tag{3.165}$$

Updates (3.165) and (3.151) make up $\mathcal{C}_{a,u,w}$ in Algorithm 2. Adding LSSC (3.153) and vertex update (3.156) to Algorithm 2 produces monotone ACGM in estimate sequence form, as listed in Algorithm 8.

Algorithm 8 Monotone ACGM in estimate sequence form
MACGM$(\boldsymbol{x}_0, L_0, A_0, \gamma_0, r_u, r_d, \mu_f, \mu_\Psi, K)$

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

3.5.3 Extrapolated Form

Monotonicity and extrapolation
In non-monotone ACGM (Algorithm 6), the auxiliary point can be obtained from two successive main iterates through extrapolation. Interestingly, this property is preserved for any value of the inverse step size. We show in the following how monotonicity alters this property and bring monotone ACGM to a form in which the auxiliary point is an extrapolation of state variables. First, we observe that extrapolation still applies to estimate sequence vertices.

Lemma 8. *The estimate sequence vertices can be obtained from state variables by extrapolation, namely*

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

Proof. See the proof of Lemma 6, with $\boldsymbol{x}_{k+1}$ replaced by $\boldsymbol{z}_{k+1}$. □

As in the non-monotone case, we define the vertex extrapolation fac-

tor sequence $\{t_k\}_{k\geq 0}$ as before using (3.125) which implies (3.126). The curvature ratio is given by (3.128) as well. We set $\boldsymbol{x}_{-1} \stackrel{\text{def}}{=} \boldsymbol{x}_0$ and $\boldsymbol{z}_0 = \boldsymbol{x}_0$.

We denote the coefficient of vertex $\boldsymbol{v}_k$ in auxiliary point update (3.162) by ω_{k+1} for all $k \geq 0$. Coefficient ω_{k+1} is therefore given by

$$\omega_{k+1} \stackrel{\text{def}}{=} \frac{a_{k+1}\gamma_k}{A_k\gamma_{k+1} + a_{k+1}\gamma_k}, \quad k \geq 0. \tag{3.166}$$

Using (3.5), (3.126), and (3.128), (3.166) becomes

$$\begin{aligned}\omega_{k+1} &= \frac{\frac{\gamma_k}{\gamma_{k+1}}}{\frac{A_{k+1}-a_{k+1}}{a_{k+1}} + \frac{\gamma_k}{\gamma_{k+1}}} = \frac{1 - q_{k+1}t_{k+1}}{t_{k+1} - 1 + 1 - q_{k+1}t_{k+1}} \\ &= \frac{1 - q_{k+1}t_{k+1}}{(1 - q_{k+1})t_{k+1}}, \quad k \geq 0. \end{aligned} \tag{3.167}$$

We can now update the auxiliary point extrapolation rule in Proposition 4 to the monotone case.

Proposition 6. *At every iteration $k \geq 0$, the new auxiliary point $\boldsymbol{y}_{k+1}$ can be obtained from the algorithm state parameters by extrapolation as*

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

where the extrapolation factor β_k is given by

$$\beta_k = (t_k - \mathbf{1}_{\{\boldsymbol{z}_k\}}(\boldsymbol{x}_k))\omega_{k+1},$$

and $\mathbf{1}_X$ denotes the membership function of set X, namely

$$\mathbf{1}_X(x) \stackrel{\text{def}}{=} \begin{cases} 1, & x \in X \\ 0, & x \notin X \end{cases}.$$

Proof. For $k = 0$, we have $\boldsymbol{z}_0 = \boldsymbol{x}_{-1} = \boldsymbol{x}_0$. Hence

$$\boldsymbol{y}_1 = \boldsymbol{x}_0 = \boldsymbol{x}_0 + \beta_0(\boldsymbol{z}_0 - \boldsymbol{x}_{-1}). \tag{3.168}$$

For $k \geq 1$, Lemma 8 combined with the auxiliary point update in (3.162) gives

$$\boldsymbol{y}_{k+1} = \frac{1}{A_k\gamma_{k+1} + a_{k+1}\gamma_k}\left(A_k\gamma_{k+1}\boldsymbol{x}_k + \frac{A_k}{a_k}\boldsymbol{z}_k + \left(a_{k+1}\gamma_k - \frac{A_k}{a_k}\right)\boldsymbol{x}_{k-1}\right). \tag{3.169}$$

Depending on the outcome of the update in line 19 of Algorithm 8 at iteration $k-1$, we distinguish two situations.

If $F(\boldsymbol{z}_k) \leq F(\boldsymbol{x}_{k-1})$, then

$$\boldsymbol{y}_{k+1} = (1 + b_k)\boldsymbol{z}_k - b_k\boldsymbol{x}_{k-1} = \boldsymbol{x}_k + b_k(\boldsymbol{z}_k - \boldsymbol{x}_{k-1}), \tag{3.170}$$

where, for brevity, we define extrapolation factor b_k as

$$b_k \stackrel{\text{def}}{=} \left(\frac{A_k}{a_k} - 1\right)\omega_{k+1}. \tag{3.171}$$

If $F(\boldsymbol{z}_k) > F(\boldsymbol{x}_{k-1})$ then, by monotonicity, we impose $\boldsymbol{x}_k = \boldsymbol{x}_{k-1}$, which leads to

$$\boldsymbol{y}_{k+1} = b'_k \boldsymbol{z}_k - (b'_k - 1)\boldsymbol{x}_{k-1} = \boldsymbol{x}_k + b'_k(\boldsymbol{z}_k - \boldsymbol{x}_{k-1}), \tag{3.172}$$

where the extrapolation factor b'_k is given by

$$b'_k \stackrel{\text{def}}{=} \left(\frac{A_k}{a_k}\right)\omega_{k+1}. \tag{3.173}$$

Expressions (3.170) and (3.172) lead to the following auxiliary point extrapolation rule:

$$\boldsymbol{y}_{k+1} = \boldsymbol{x}_k + \beta_k(\boldsymbol{z}_k - \boldsymbol{x}_{k-1}), \tag{3.174}$$

where

$$\beta_k = \begin{cases} b_k, & \boldsymbol{x}_k = \boldsymbol{z}_k \\ b'_k, & \boldsymbol{x}_k = \boldsymbol{x}_{k-1} \end{cases}. \tag{3.175}$$

Auxiliary point extrapolation factor β_k can be written as

$$\begin{aligned} \beta_k &= \begin{cases} (t_k - 1)\omega_{k+1}, & \boldsymbol{x}_k = \boldsymbol{z}_k \\ t_k\omega_{k+1}, & \boldsymbol{x}_k = \boldsymbol{x}_{k-1} \end{cases}, \\ &= (t_k - \mathbf{1}_{\{\boldsymbol{z}_k\}}(\boldsymbol{x}_k))\omega_{k+1}. \end{aligned} \tag{3.176}$$

□

We simplify monotone ACGM further by noting that, to produce the auxiliary point, the extrapolation rule in Proposition 6 depends on three vector parameters. However, it is not necessary to store both $\boldsymbol{z}_k$ and $\boldsymbol{x}_{k-1}$ across iterations. To address applications where memory is limited, we only maintain the difference term $\boldsymbol{d}_k$, given by

$$\boldsymbol{d}_k = (t_k - \mathbf{1}_{\{\boldsymbol{z}_k\}}(\boldsymbol{x}_k))(\boldsymbol{z}_k - \boldsymbol{x}_{k-1}), \qquad k \geq 0. \tag{3.177}$$

The difference term in (3.177) simplifies Proposition 6 to

$$\boldsymbol{y}_{k+1} = \boldsymbol{x}_k + \omega_{k+1}\boldsymbol{d}_k, \qquad k \geq 0. \tag{3.178}$$

The above modifications yield a form of monotone ACGM (MACGM) based on extrapolation, which we list in Algorithm 9.

Note that subexpression ω_{k+1} contains only recent information whereas $\boldsymbol{d}_k$ needs only to access the state of the preceding iteration.

We stress that while Algorithms 8 and 9 carry out different computations, they are mathematically equivalent with respect to the main iterate sequence $\{\boldsymbol{x}_k\}_{k\geq 0}$.

Lemma 7 applies to monotone ACGM as well. Consequently, when $\gamma_0 \neq A_0\mu$, the convergence guarantee can be obtained from the state parameters.

Algorithm 9 Monotone ACGM (MACGM) in extrapolated form
MACGM($\boldsymbol{x}_0, L_0, A_0, \gamma_0, r_u, r_d, \mu_f, \mu_\Psi, K$)

$\boldsymbol{x}_{-1} = \boldsymbol{x}_0$
$\boldsymbol{d}_0 = \boldsymbol{0}$
$\mu = \mu_f + \mu_\Psi$
$t_0 = \sqrt{\frac{(L_0+\mu_\Psi)A_0}{\gamma_0}}$
$q_0 = \frac{\mu}{L_0+\mu_\Psi}$
for $k = 0, ..., K-1$ **do**
  $\hat{L}_{k+1} := r_d L_k$
  loop
    $\hat{q}_{k+1} := \frac{\mu}{\hat{L}_{k+1}+\mu_\Psi}$
    $\hat{t}_{k+1} := \frac{1}{2}\left(1 - q_k t_k^2 + \sqrt{(1-q_k t_k^2)^2 + 4\frac{\hat{L}_{k+1}+\mu_\Psi}{L_k+\mu_\Psi} t_k^2}\right)$
    $\hat{\boldsymbol{y}} := \boldsymbol{x}_k + \frac{1-\hat{q}_{k+1}\hat{t}_{k+1}}{(1-\hat{q}_{k+1})\hat{t}_{k+1}}\boldsymbol{d}_k$
    $\hat{\boldsymbol{z}} := \mathrm{prox}_{\frac{1}{\hat{L}_{k+1}}\Psi}\left(\hat{\boldsymbol{y}} - \frac{1}{\hat{L}_{k+1}}\nabla f(\hat{\boldsymbol{y}})\right)$
    if $f(\hat{\boldsymbol{z}}) \le Q_{f,\hat{L}_{k+1},\hat{\boldsymbol{y}}}(\hat{\boldsymbol{z}})$ **then**
      Break from loop
    else
      $\hat{L}_{k+1} := r_u \hat{L}_{k+1}$
    end if
  end loop
  $L_{k+1} := \hat{L}_{k+1},\ q_{k+1} := \hat{q}_{k+1},\ t_{k+1} := \hat{t}_{k+1}$
  $\boldsymbol{z}_{k+1} := \hat{\boldsymbol{z}}_{k+1}$
  $\boldsymbol{x}_{k+1} = \arg\min\{F(\boldsymbol{z}_{k+1}), F(\boldsymbol{x}_k)\}$
  $\boldsymbol{d}_{k+1} = (t_{k+1} - \boldsymbol{1}_{\{\boldsymbol{z}_{k+1}\}}(\boldsymbol{x}_{k+1}))(\boldsymbol{z}_{k+1} - \boldsymbol{x}_k)$
end for

Border-case

When $\gamma_0 = A_0\mu$, (3.139) holds as is. In this border-case, Algorithm 9 can be simplified further. It follows from (3.139) that the auxiliary point extrapolation factor is given by

$$\beta_k = \frac{\sqrt{L_k+\mu_\Psi} - \boldsymbol{1}_{\{\boldsymbol{z}_k\}}(\boldsymbol{x}_k)\sqrt{\mu}}{\sqrt{L_{k+1}+\mu_\Psi} + \sqrt{\mu}}, \qquad k \ge 0. \tag{3.179}$$

The convergence guarantee recursion rule in (3.147) remains valid and we also choose $A_0 = 1$ and $\gamma_0 = \mu$. To reduce computational intensity, we modify subexpressions $\boldsymbol{d}_k$ and ω_{k+1} as

$$\boldsymbol{d}_k = \left(\sqrt{L_k+\mu_\Psi} - \boldsymbol{1}_{\{\boldsymbol{z}_k\}}(\boldsymbol{x}_k)\sqrt{\mu}\right)(\boldsymbol{z}_k - \boldsymbol{x}_{k-1}), \quad k \ge 0, \tag{3.180}$$

$$\omega_{k+1} = \frac{1}{\sqrt{L_{k+1}+\mu_\Psi} + \sqrt{\mu}}, \quad k \ge 0. \tag{3.181}$$

This simpler form taken by border-case monotone ACGM (BMACGM) is listed in Algorithm 10.

Algorithm 10 Border-case Monotone ACGM (BMACGM) in extrapolated form

BMACGM($\boldsymbol{x}_0, L_0, r_u, r_d, \mu_f, \mu_\Psi, K$**)**

$\boldsymbol{x}_{-1} = \boldsymbol{x}_0$
$\boldsymbol{d}_0 = \mathbf{0}$
$\mu = \mu_f + \mu_\Psi$
$A_0 = 1$
for $k = 0, ..., K-1$ **do**
  $\hat{L}_{k+1} := r_d L_k$
  loop
    $\hat{\boldsymbol{y}}_{k+1} := \boldsymbol{x}_k + \frac{1}{\sqrt{\hat{L}_{k+1}+\mu_\Psi}+\sqrt{\mu}}\boldsymbol{d}_k$
    $\hat{\boldsymbol{z}}_{k+1} := \operatorname{prox}_{\frac{1}{\hat{L}_{k+1}}\Psi}\left(\hat{\boldsymbol{y}}_{k+1} - \frac{1}{\hat{L}_{k+1}}\nabla f(\hat{\boldsymbol{y}}_{k+1})\right)$
    if $f(\hat{\boldsymbol{z}}_{k+1}) \leq Q_{f,\hat{L}_{k+1},\hat{\boldsymbol{y}}_{k+1}}(\hat{\boldsymbol{z}}_{k+1})$ **then**
      Break from loop
    else
      $\hat{L}_{k+1} := r_u \hat{L}_{k+1}$
    end if
  end loop
  $L_{k+1} := \hat{L}_{k+1}$
  $\boldsymbol{z}_{k+1} := \hat{\boldsymbol{z}}_{k+1}$
  $\boldsymbol{x}_{k+1} = \arg\min\{F(\boldsymbol{z}_{k+1}), F(\boldsymbol{x}_k)\}$
  $\boldsymbol{d}_{k+1} = \left(\sqrt{L_{k+1}+\mu_\Psi} - \mathbf{1}_{\{\boldsymbol{z}_{k+1}\}}(\boldsymbol{x}_{k+1})\sqrt{\mu}\right)(\boldsymbol{z}_{k+1} - \boldsymbol{x}_k)$
  $A_{k+1} = \frac{\sqrt{L_{k+1}+\mu_\Psi}}{\sqrt{L_{k+1}+\mu_\Psi}-\sqrt{\mu}}A_k$
end for

4. Analysis of ACGM

4.1 Revisiting the Estimate Sequence

In Section 2.4, we have seen how the estimate sequence can be derived as a substitute convergence guarantee and how the estimate sequence property in (2.15) is a sufficient condition for the convergence guarantees in the ISDUB expression (2.6).

Let the estimate sequence gap be defined as

$$\tilde{\Gamma}_k \stackrel{\text{def}}{=} A_k F(\boldsymbol{x}_k) - \psi_k^*, \quad k \geq 0. \tag{4.1}$$

FGM was designed in [16] to maintain the Lyapunov property of the estimate sequence gap, stated as

$$\tilde{\Gamma}_{k+1} \leq \tilde{\Gamma}_k, \quad k \geq 0. \tag{4.2}$$

The structure of the initial estimate function in (2.11), which applies to FGM as well, implies that $\tilde{\Gamma}_0 = 0$ for any initial parameter choice and therefore (4.2) constitutes a sufficient condition for the estimate sequence property (2.15) in FGM.

In this section, we investigate whether (4.2) applies to any variant of ACGM and if it can also be used to construct ACGM[1]. First, note that the estimate sequence gap terms in (4.1) involve the optimal values of the estimate functions. These values do not depend on unknown quantities and can be updated as ACGM progresses.

Lemma 9. *For all variants of ACGM, the estimate function optimal values can be updated at every iteration $k \geq 0$ as*

$$\psi_{k+1}^* = \psi_k^* + a_{k+1} F(\boldsymbol{x}_{k+1}) + \left(\frac{a_{k+1}}{2L'_{k+1}} - \frac{a_{k+1}^2}{2\gamma_{k+1}} \right) \| g_{L'_{k+1}}(\boldsymbol{y}_{k+1}) \|_2^2 + \frac{a_{k+1}\gamma_k}{\gamma_{k+1}} \left(\left\langle g_{L'_{k+1}}(\boldsymbol{y}_{k+1}), \boldsymbol{y}_{k+1} - \boldsymbol{v}_k \right\rangle + \frac{\mu}{2} \| \boldsymbol{v}_k - \boldsymbol{y}_{k+1} \|_2^2 \right).$$

[1] We thank Rose Sfeir for pointing out Subsection 2.2.4 in [16].

Proof. The derivation style proposed in [16] considerably reduces the number of calculations required to reach the solution by exploiting the fact that the lower bounds are formulated around the auxiliary point. Since we have used the same strategy when designing the lower bounds of ACGM, for the sake of brevity, we construct this proof along the lines of [16]. Every result in this proof holds for all $k \geq 0$.

From (2.46) and (3.7), we have at point $\boldsymbol{y}_{k+1}$ that

$$\psi_{k+1}(\boldsymbol{y}_{k+1}) = \psi^*_{k+1} + \frac{\gamma_{k+1}}{2}\|\boldsymbol{y}_{k+1} - \boldsymbol{v}_{k+1}\|_2^2 = \psi_k(\boldsymbol{y}_{k+1}) + a_{k+1}w_{k+1}(\boldsymbol{y}_{k+1}). \tag{4.3}$$

By expanding $\psi_k(\boldsymbol{y}_{k+1})$ in (4.3) using (2.46), we obtain an estimate function optimal value update rule as

$$\psi^*_{k+1} = \psi^*_k + \frac{\gamma_k}{2}\|\boldsymbol{y}_{k+1} - \boldsymbol{v}_k\|_2^2 - \frac{\gamma_{k+1}}{2}\|\boldsymbol{y}_{k+1} - \boldsymbol{v}_{k+1}\|_2^2 + a_{k+1}w_{k+1}(\boldsymbol{y}_{k+1}). \tag{4.4}$$

Note that all variants of ACGM introduced in this work share the same simple lower bound expression, given by (3.80). Expanding $w_{k+1}(\boldsymbol{y}_{k+1})$ using (3.80) yields

$$\begin{aligned}\psi^*_{k+1} = \psi^*_k + a_{k+1}F(\boldsymbol{x}_{k+1}) + \frac{a_{k+1}}{2L'_{k+1}}\|g_{L'_{k+1}}(\boldsymbol{y}_{k+1})\|_2^2 \\ + \frac{\gamma_k}{2}\|\boldsymbol{y}_{k+1} - \boldsymbol{v}_k\|_2^2 - \frac{\gamma_{k+1}}{2}\|\boldsymbol{y}_{k+1} - \boldsymbol{v}_{k+1}\|_2^2.\end{aligned} \tag{4.5}$$

Combining (3.82) and (3.83) gives

$$\boldsymbol{y}_{k+1} - \boldsymbol{v}_{k+1} = \frac{\gamma_k}{\gamma_{k+1}}(\boldsymbol{y}_{k+1} - \boldsymbol{v}_k) + \frac{a_{k+1}}{\gamma_{k+1}}g_{L'_{k+1}}(\boldsymbol{y}_{k+1}). \tag{4.6}$$

The simple expression in (4.6) leads to

$$\begin{aligned}\frac{\gamma_{k+1}}{2}\|\boldsymbol{y}_{k+1} - \boldsymbol{v}_{k+1}\|_2^2 = \frac{a_{k+1}^2}{2\gamma_{k+1}}\|g_{L'_{k+1}}(\boldsymbol{y}_{k+1}\|^2 \\ + \frac{a_{k+1}\gamma_k}{\gamma_{k+1}}\left\langle g_{L'_{k+1}}(\boldsymbol{y}_{k+1}), \boldsymbol{y}_{k+1} - \boldsymbol{v}_k\right\rangle + \frac{\gamma_k^2}{2\gamma_{k+1}}\|\boldsymbol{y}_{k+1} - \boldsymbol{v}_k)\|^2.\end{aligned} \tag{4.7}$$

We substitute (4.7) in (4.5) and use (3.82) to get the desired result. □

Given the generality of Lemma 9, in the remainder of this section we will use the term ACGM to denote all variants of ACGM introduced in this work, collectively.

Theorem 6. *In ACGM, if at iteration $k \geq 0$ the descent condition for f in* (3.15) *holds, then*

$$\tilde{\Gamma}_{k+1} + \mathcal{A}_{k+1} + \frac{1}{\gamma_{k+1}}\left\langle g_{L'_{k+1}}(\boldsymbol{y}_{k+1}), \boldsymbol{s}_{k+1}\right\rangle \leq \tilde{\Gamma}_k,$$

where subexpression $\mathcal{A}_{k+1}$ and $\boldsymbol{s}_{k+1}$ are given by (3.85) *and* (3.87)*, respectively, without modification.*

Proof. Using (3.5), (4.1), and Lemma 9 we obtain

$$-\tilde{\Gamma}_{k+1} = -\tilde{\Gamma}_k + A_k(F(\boldsymbol{x}_k) - F(\boldsymbol{x}_{k+1})) + \left(\frac{a_{k+1}}{2L'_{k+1}} - \frac{a_{k+1}^2}{2\gamma_{k+1}}\right)\|g_{L'_{k+1}}(\boldsymbol{y}_{k+1})\|_2^2 + \frac{1}{\gamma_{k+1}}\left(\left\langle g_{L'_{k+1}}(\boldsymbol{y}_{k+1}), a_{k+1}\gamma_k(\boldsymbol{y}_{k+1} - \boldsymbol{v}_k)\right\rangle + \frac{\mu}{2}\|\boldsymbol{v}_k - \boldsymbol{y}_{k+1}\|_2^2\right), \quad k \geq 0. \tag{4.8}$$

Since the descent condition in (3.15) is satisfied, then Proposition 3 holds and, by multiplying both sides of the inequality in Proposition 3 at $\boldsymbol{x}_k$ with non-negative quantity A_k, we have that

$$A_k(F(\boldsymbol{x}_k) - F(\boldsymbol{x}_{k+1})) \geq \frac{A_k}{2L_{k+1}}\|g_{L'_{k+1}}(\boldsymbol{y}_{k+1})\|_2^2 + \left\langle g_{L'_{k+1}}(\boldsymbol{y}_{k+1}), A_k(\boldsymbol{x}_k - \boldsymbol{y}_{k+1})\right\rangle + \frac{A_k\mu}{2}\|\boldsymbol{x}_k - \boldsymbol{y}_{k+1}\|_2^2, \quad k \geq 0. \tag{4.9}$$

Adding together (4.8) and (4.9) yields

$$-\tilde{\Gamma}_{k+1} \geq -\tilde{\Gamma}_k + \mathcal{A}_{k+1} + \frac{1}{\gamma_{k+1}}\left\langle g_{L'_{k+1}}(\boldsymbol{y}_{k+1}), \boldsymbol{s}_{k+1}\right\rangle + \frac{\mu}{2\gamma_{k+1}}\left(A_k\gamma_{k+1}\|\boldsymbol{x}_k - \boldsymbol{y}_{k+1}\|_2^2 + a_{k+1}\gamma_k\|\boldsymbol{v}_k - \boldsymbol{y}_{k+1}\|_2^2\right), \quad k \geq 0. \tag{4.10}$$

The square terms $\|\boldsymbol{x}_k - \boldsymbol{y}_{k+1}\|_2^2$ and $\|\boldsymbol{v}_k - \boldsymbol{y}_{k+1}\|_2^2$ in (4.10) are non-negative and can be left out. □

Theorem 6 provides a means of satisfying the Lyapunov property of the estimate sequence in (4.2) for any algorithmic state of ACGM. Namely, for the same reasons outlined in Chapter 3, we need to ensure that $\mathcal{A}_{k+1} \geq 0$ and $\boldsymbol{s}_{k+1} = \boldsymbol{0}$. These two conditions are identical to the ones we have derived from Theorems 4 and 5.

This observation has two implications. First, the original estimate sequence property in (2.15) holds for ACGM, regardless of the algorithmic state. Second, ACGM can be derived using the non-augmented estimate sequence. Thus, in the context of ACGM, the estimate sequence and the augmented estimate sequence can be used interchangeably. Each approach comes with its own advantages. The numerical value of the estimate sequence gap in (4.1) can be computed at every iteration. On the other hand, the gap sequence that results from the augmented estimate sequence has a simple closed form expression and effectively bridges the concepts of estimate sequence and Lyapunov functions. Moreover, the gap sequence terms in (2.55) contain both an image space term and a domain space term, whereas the estimate sequence gap in (4.1) is an image space only term. This special structure of the gap sequence can be used, for instance, to study the convergence of the iterates (see [26]).

We leave a detailed comparison of the two approaches, along with their wider implications, as a topic for future research.

4.2 Worst-case Convergence Guarantees

Regardless of whether we use the estimate sequence or the augmented estimate sequence, every variant of ACGM introduced in this work adheres to the design pattern in Algorithm 2, with the iterates obeying the ISDUB expression in (2.6). As such, the state variable A_k, whether maintained explicitly or derived from other state parameters, gives the convergence guarantee at the beginning of iteration $k \geq 0$. Note that non-strongly convex ACGM is a particular case of ACGM for arbitrary strong convexity and the convergence guarantees of non-monotone ACGM are identical to those of monotone ACGM by virtue of Theorem 4. Thus, the results in this section, as in Section 4.1, apply to all versions of ACGM, unless stated otherwise.

At every iteration $k \geq 0$, the larger the LCE L_{k+1}, the slower the algorithm. Therefore, we need to establish an upper bound for L_{k+1} that would enable us to formulate worst-case convergence guarantees.

Lemma 10. *At every iteration $k \geq 0$ of ACGM, the LCE is upper bounded as*

$$L_{k+1} \leq L_u,$$

where the worst-case Lipschitz constant estimate L_u is given by

$$L_u \stackrel{\text{def}}{=} \max\{r_u L_f, r_d L_0\}.$$

Proof. Let $\tilde{k}$ denote the number of consecutive iterations, starting at $k = 0$, for which no backtracks occur. Note that the value of $\tilde{k}$ can be zero. We have that

$$L_k = r_d^k L_0, \quad 0 \leq k \leq \tilde{k}. \tag{4.11}$$

Therefore

$$L_{k+1} = r_d^{k+1} L_0 < r_d L_0, \quad 0 \leq k \leq \tilde{k} - 1. \tag{4.12}$$

The definition of $\tilde{k}$ implies that the first candidate $\hat{L}_{\tilde{k}+1}$ will fail the LSSC. ACGM will backtrack until it outputs an estimate $L_{\tilde{k}+1}$ such that $L_{\tilde{k}+1}/r_u$ fails and $L_{\tilde{k}+1}$ passes. Hence, $L_{\tilde{k}+1}/r_u < L_f$, which implies that

$$L_{\tilde{k}+1} < r_u L_f. \tag{4.13}$$

We prove by induction that

$$L_{k+1} < r_u L_f, \quad k \geq \tilde{k}. \tag{4.14}$$

First, (4.13) shows that (4.14) holds for $k = \tilde{k}$. Next, we assume (4.14) holds for a certain $k \geq \tilde{k}$ and show that (4.14) applies to $k + 1$ as well. At iteration k, we distinguish two scenarios. If no backtrack occurs, then we have

$$L_{k+1} = r_d L_k < L_k < r_u L_f. \tag{4.15}$$

If one or more backtracks occur, following the same reasoning as in iteration $\tilde{k}$, we have that

$$L_{k+1}/r_u < L_f. \tag{4.16}$$

Thus, (4.14) is valid. Combining (4.12) with (4.14) gives

$$L_{k+1} < \max\{r_u L_f, r_d L_0\}, \quad k \geq 0. \tag{4.17}$$

□

For our analysis, we will also need to define the worst-case local inverse condition number q_u, given by

$$q_u \overset{\text{def}}{=} \frac{\mu}{L_u + \mu_\Psi}. \tag{4.18}$$

Using Lemma 10 and (4.18), we can state the worst-case convergence guarantees of ACGM.

Theorem 7. *If $\gamma_0 \geq A_0\mu$, ACGM generates a sequence $\{\boldsymbol{x}_k\}_{k\geq 1}$ that satisfies*

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

where the normalized gap term $\bar{\Delta}_0$ is given by

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

Proof. The curvature γ_k can be lower bounded based on (3.84) as

$$\gamma_k = \gamma_0 + \mu(A_k - A_0) \geq \gamma_0, \quad k \geq 0. \tag{4.19}$$

This bound is accurate when $\mu = 0$ or the inverse condition number is negligibly low, namely $q \ll 1/K$, where $K \geq 1$ is the total number of iterations performed by ACGM (see Section 3.1).

From (3.5), (3.120), and (4.19) we have that

$$\begin{aligned} A_{k+1} &= A_k + \mathcal{E}(\gamma_k, A_k, L_{k+1}) \\ &\geq A_k + \frac{\gamma_0}{2(L_{k+1} - \mu_f)} + \sqrt{\frac{\gamma_0^2}{4(L_{k+1} - \mu_f)^2} + \frac{A_k\gamma_0}{(L_{k+1} - \mu_f)}}, \quad k \geq 0. \end{aligned} \tag{4.20}$$

Regardless of the outcome of individual line-search calls, Lemma 10 guarantees that

$$\begin{aligned} A_{k+1} &\geq A_k + \frac{\gamma_0}{2(L_u - \mu_f)} + \sqrt{\frac{\gamma_0^2}{4(L_u - \mu_f)^2} + \frac{A_k\gamma_0}{(L_u - \mu_f)}} \\ &= A_k + \frac{\gamma_0}{L_u - \mu_f}\left(\frac{1}{2} + \sqrt{\frac{1}{4} + \frac{A_k(L_u - \mu_f)}{\gamma_0}}\right), \quad k \geq 0. \end{aligned} \tag{4.21}$$

To simplify the derivation, we define the scaled convergence guarantee $\tilde{A}_k$ as

$$\tilde{A}_k \stackrel{\text{def}}{=} \frac{A_k(L_u - \mu_f)}{\gamma_0}, \quad k \geq 0. \tag{4.22}$$

Using (4.22) in (4.21), we obtain

$$\tilde{A}_{k+1} \geq \tilde{A}_k + \frac{1}{2} + \sqrt{\frac{1}{4} + \tilde{A}_k}, \quad k \geq 0. \tag{4.23}$$

We show by induction that

$$\tilde{A}_k \geq \frac{(k+1)^2}{4}, \quad k \geq 1. \tag{4.24}$$

First, for $k = 1$, $\tilde{A}_0 \geq 0$ in (4.23) implies that

$$\tilde{A}_1 \geq \tilde{A}_0 + \frac{1}{2} + \sqrt{\frac{1}{4} + \tilde{A}_0} \geq \frac{1}{2} + \sqrt{\frac{1}{4}} = 1 = \frac{(1+1)^2}{4}. \tag{4.25}$$

Next, we show that if (4.24) holds for a certain $k \geq 1$ then it does for $k+1$ as well.

$$\begin{aligned} \tilde{A}_{k+1} &\geq \tilde{A}_k + \frac{1}{2} + \sqrt{\frac{1}{4} + \tilde{A}_k} \geq \frac{(k+1)^2}{4} + \frac{1}{2} + \sqrt{\frac{1}{4} + \frac{(k+1)^2}{4}} \\ &> \frac{1}{4}\left(k^2 + 2k + 1 + 2 + 2\sqrt{(k+1)^2}\right) = \frac{1}{4}(k^2 + 4k + 5) \\ &> \frac{(k+2)^2}{4}. \end{aligned} \tag{4.26}$$

Combining ISDUB expression (2.6) with (4.24) yields

$$F(\boldsymbol{x}_k) - F(\boldsymbol{x}^*) \leq \frac{4}{(k+1)^2}(L_u - \mu_f)\bar{\Delta}_0, \quad k \geq 1. \tag{4.27}$$

We have from (3.115) and assumption $\gamma_0 \geq A_0\mu$ that

$$\begin{aligned} \frac{a_{k+1}^2}{A_{k+1}^2} &= \frac{\gamma_{k+1}}{(L_{k+1} + \mu_\Psi)A_{k+1}} = \frac{(\gamma_0 - A_0\mu) + A_{k+1}\mu}{(L_{k+1} + \mu_\Psi)A_{k+1}} \geq q_{k+1} \\ &\geq q_u, \quad k \geq 0, \end{aligned} \tag{4.28}$$

which, combined with $q_u \geq 0$, implies that

$$\frac{A_{k+1}}{A_k} \geq \frac{1}{\sqrt{1 - q_u}}, \quad k \geq 0. \tag{4.29}$$

This bound is meaningful in tracking the progress of ACGM only when $\mu > 0$ and q is of a magnitude comparable to that of $1/K$. Under such conditions, we can obtain a simple lower bound on A_1 by using $A_0 \geq 0$ in (3.120) in the form of

$$A_1 = A_0 + \mathcal{E}(\gamma_0, A_0, L_1) \geq \frac{\gamma_0 + \sqrt{\gamma_0^2}}{2(L_1 - \mu_f)} \geq \frac{\gamma_0}{L_u - \mu_f}. \tag{4.30}$$

By iterating (4.29) starting at (4.30), we obtain

$$A_{k+1} \geq \frac{\gamma_0}{(L_u - \mu_f)(1 - \sqrt{q_u})^k}, \quad k \geq 0. \tag{4.31}$$

Substituting (4.31) in the ISDUB expression (2.6) yields

$$F(\boldsymbol{x}_k) - F(\boldsymbol{x}^*) \leq (1 - \sqrt{q_u})^{k-1}(L_u - \mu_f)\bar{\Delta}_0, \quad k \geq 1. \tag{4.32}$$

The desired result is the combination of (4.27) and (4.32). □

Note that the assumption $\gamma_0 \geq A_0\mu$ always holds for non-strongly convex objectives and that ACGM is guaranteed to converge for strongly convex objectives also when $\gamma_0 < A_0\mu$ (see Subsection 3.4.3). However, in the latter case, it is more difficult to obtain simple lower bounds on the convergence guarantees. We leave such an endeavor to future research.

4.3 Wall-clock Time Units

So far, we have measured the theoretical performance of algorithms in terms of convergence guarantees (including the worst-case ones) indexed in iterations. This does not account for the complexity of individual iterations. From now on, we distinguish between two types of convergence guarantees. One is the aforementioned *iteration convergence guarantee* and a new *computational convergence guarantee*, introduced in the sequel.

In the literature, the prevailing indexing strategies for objective values used in measuring the convergence rate are based on either iterations (e.g., [6, 17–19]), running time in a particular computing environment (e.g., [6, 19]), or the number of calls to a low-level routine that dominates all others in complexity (e.g., [17, 36]). The first approach, which we have also used in the derivation of ACGM, cannot cope with the diversity of methods studied. For example, when no backtracks occur, AMGS makes two calls to $\nabla f(\boldsymbol{x})$ 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 [37].

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 [37]. 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[2], 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. The computing system is further assumed to be uniform memory access (UMA) [37] and to be able to store the arguments and the results of oracle calls in memory for as long as they are needed.

Thus, the computational convergence rate is given by the objective distance decrease rate indexed in WTU, when the optimization algorithm is executed on the aforementioned shared memory parallel system.

4.3.1 Standard WTU

To account for the broadness of the problem class, wherein oracle functions may or may not be separable[3] and their relative cost may vary, we impose that the complexity of computing $f(\boldsymbol{x})$ is comparable to that of $\nabla f(\boldsymbol{x})$ [23]. 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 element-wise vector operations, including scalar-vector multiplications, vector additions, and inner products [4].

Non-monotone ACGM

In the following, we analyze the resource usage and runtime behavior of FGM, AMGS, FISTA, FISTA-CP, and non-monotone ACGM under the above assumptions. We will argue in Section 4.4 why we do not need to consider other methods applicable to composite problems.

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.

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

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

For methods that employ line-search, backtracks stall the algorithm in a way that cannot be alleviated by parallelization or intensity reduction. Therefore, it is desirable to reduce their frequency. Assuming that the local curvature of f varies around a fixed value, the likelihood of a backtrack occurring is given by $-log(r_d)/\log(r_u)$. Therefore, $\log(r_u)$ should be significantly larger than $-log(r_d)$. With such a parameter choice, the algorithm can proceed from one iteration to another by speculating that backtracks do not occur at all. This technique is referred to in the literature as speculative execution [37] 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.

AMGS requires at iteration k calls to both $\nabla f(\boldsymbol{y}_{k+1})$ and $\nabla f(\boldsymbol{x}_{k+1})$. The 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 $\nabla f(\boldsymbol{y}_{k+1})$, followed by $\nabla f(\boldsymbol{x}_{k+1})$, 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 LCE. The $f(\boldsymbol{y}_{k+1})$ call can be performed in parallel 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 LCE necessitates the recalculation of $\boldsymbol{y}_{k+1}$, in addition to the delay in the backtrack condition assessment. As a result, non-monotone ACGM has an iteration base cost of 1 WTU and a 2 WTU backtrack cost. The Algorithms 5, 6, and 7 forms of ACGM (along with particular cases Algorithms 3 and 4, respectively) are identical with respect to WTU usage.

The iteration costs of AMGS, FISTA, and non-monotone ACGM are summarized in Table 4.1. Note that, with properly tuned parameters, as explained above, iterations without backtracks are by far the most common. Hence, the backtrack costs have a negligible impact on algorithm performance.

Interestingly, when using standard WTU, 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 iterations these computations are part of, are detailed in Table 4.2 for an iteration $k \geq 1$ without backtracks and in Table 4.3 for an iteration where a single backtrack occurs. The behavior of subsequent

Table 4.1. Per iteration cost in WTU of line-search methods AMGS, FISTA, and non-monotone ACGM

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

backtracks follows closely the pattern shown in Table 4.3.

Table 4.2. Resource allocation and runtime behavior of parallel black-box FGM, FISTA-CP, AMGS, FISTA, and non-monotone 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

Monotone ACGM

We define an overshoot of monotone ACGM at iteration $k \geq 0$ as the event in which the new iterate has a larger objective value than the previous one, i.e., the monotone condition (MC), given by $F(\boldsymbol{x}_{k+1}) \leq F(\boldsymbol{x}_k)$, fails. We assume that overshoots occur even less frequently than backtracks. Similarly, the algorithm can proceed by speculating that the MC always passes and defaults to $\boldsymbol{x}_{k+1} = \boldsymbol{z}_{k+1}$. The monotone condition can be computed at the same time as the LSSC and does not incur additional computation. Therefore, the MC and the LSSC can be fused into a single condition. If the LSSC fails, the MC is not evaluated. This leaves three possible outcomes, outlined in Table 4.4.

Table 4.3. Resource allocation and runtime behavior of parallel black-box AMGS, FISTA, and non-monotone 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

Table 4.4. Additional cost in WTU incurred by the fused LSSC / MC condition in MACGM

	MC passed	MC failed
LSSC passed	0	1
LSSC failed	2	N / A

4.3.2 Generalized WTU

In order to compare algorithms based on a unified benchmark, we have assumed in Subsection 4.3.1 that $f(\boldsymbol{x})$ and $\nabla f(\boldsymbol{x})$ require 1 WTU each while all other operations are negligible and amount to 0 WTU. In this section, we generalize the analysis. We attribute finite non-negative costs t_f, t_g, t_Ψ, and t_p to $f(\boldsymbol{x})$, $\nabla f(\boldsymbol{x})$, $\Psi(\boldsymbol{x})$, and $\operatorname{prox}_{\tau\Psi}(\boldsymbol{x})$, respectively. However, since we are dealing with large-scale problems, we maintain the assumption that element-wise vector operations have negligible complexity when compared to oracle functions and assign a cost of 0 WTU to each. Synchronization of PPUs also incurs no cost. Consequently, when computed in isolation, an objective function value $F(\boldsymbol{x})$ call costs $t_F = \max\{t_f, t_\Psi\}$, ascribable to separability, while a proximal gradient operation costs $t_T = t_g + t_p$, due to computational dependencies.

Non-monotone ACGM

The advancement phase of an ACGM iteration consists of one proximal gradient step. Hence, every iteration has a base cost of $t_T = t_g + t_p$.

If the LSSC of iteration $k \geq 0$ fails, then the algorithm discards all the state information pertaining to all iterations made after k, reverts to iteration k, and performs the necessary computations to correct the error. We consider that a mis-prediction incurs a detection cost t_d^{LSSC} and a correction cost t_c^{LSSC}. The LSSC requires the evaluation of $f(\boldsymbol{z}_{k+1})$ and incurs a detection cost of $t_d^{\mathrm{LSSC}} = t_f$. A backtrack entails recomputing $\boldsymbol{y}_{k+1}$, yielding a LSSC correction time of $t_c^{\mathrm{LSSC}} = t_T$. Thus, for non-monotone ACGM, each backtrack adds $t_f + t_T$ WTU to a base iteration cost of t_T. A comparison to other methods employing line-search is shown in Table 4.5.

Table 4.5. Per-iteration cost of FISTA, AMGS, and non-monotone ACGM

	FISTA	AMGS	ACGM
Base cost	$t_g + t_p$	$2t_g + 2t_p$	$t_g + t_p$
t_d^{LSSC}	t_f	t_g	t_f
t_c^{LSSC}	t_p	$t_g + t_p$	$t_g + t_p$
Backtrack cost	$t_f + t_p$	$2t_g + t_p$	$t_f + t_g + t_p$

Monotone ACGM

The LSSC and the MC can be evaluated in parallel with subsequent iterations. Both rely on the computation of $f(\boldsymbol{z}_{k+1})$, which in the worst case requires $\lceil t_f / t_T \rceil$ dedicated PPUs. In addition, the MC may need up to $\lceil t_\Psi / t_T \rceil$ PPUs.

Monotone ACGM proceeds speculating that the MC always passes. Hence, the MC has $t_d^{\mathrm{MC}} = t_F$, due to its dependency on $\Psi(\boldsymbol{z}_{k+1})$, but once the algorithmic state of iteration k has been restored, no additional oracle calls are needed, leading to $t_c^{\mathrm{MC}} = 0$. The possible outcomes of the fused LSSC / MC condition and their cost in generalized WTU are listed in Table 4.6.

Table 4.6. Monotone ACGM stall time in generalized WTU based on the outcome of the fused LSSC / MC condition

	MC passed	MC failed
LSSC passed	0	$\max\{t_f, t_\Psi\}$
LSSC failed	$t_f + t_g + t_p$	N / A

4.4 ACGM among its Class of Algorithms

4.4.1 Uniting Nesterov's FGM and FISTA

Due to its generality, ACGM is able to incorporate, as particular instances, both FGM and FISTA, in their most common forms.

ACGM, when formulated using extrapolation (in particular, the versions listed in Algorithms 4, 6, 7, and 9), encompasses several variants of FISTA. Specifically, non-monotone ACGM (Algorithm 6) without the line-search procedure, where

$$L_k = L_f, \quad k \geq 0, \tag{4.33}$$

produces the same iterates as FISTA-CP [6] with the theoretically optimal step size

$$\tau^{\mathrm{FISTA-CP}} = \frac{1}{L_f}. \tag{4.34}$$

Monotone ACGM (Algorithm 9) without line search is equivalent to the monotone variant of FISTA-CP with the optimal step size in (4.34). For a special subclass of composite problems with non-strongly convex regularizers ($\mu_\Psi = 0$ and $\mu = \mu_f > 0$), border-case non-monotone ACGM (Algorithm 7) with line-search matches the scAPG method introduced in [21].

In the non-strongly convex case, Algorithm 4 without line-search coincides with the original formulation of constant step size FISTA in [5]. Adding monotonicity (Algorithm 9 with $\mu = \mu_f = \mu_\Psi = 0$ and fixed step size) yields monotone FISTA (MFISTA) in [22]. Also for $\mu = 0$, ACGM with line-search (Algorithm 4) is a generalization (whereby A_0 need not be zero) of the robust FISTA-like method described in [38] which in turn constitutes a simplification of a recently introduced line-search extension of FISTA [39].

When dealing with differentiable objectives, we can assume without loss of generality that

$$\Psi(\boldsymbol{x}) = 0, \quad \boldsymbol{x} \in \mathbb{R}^n. \tag{4.35}$$

In this context, non-monotone ACGM in estimate sequence form (Algorithm 5) without line search has the local upper bounds given by $Q_{f,L_f,\boldsymbol{y}_{k+1}}(\boldsymbol{x})$ at every iteration $k \geq 0$. By substituting these local upper bound functions with any functions $u_{k+1}(\boldsymbol{x})$ that produce iterates satisfying the descent condition, which means in this context that

$$\begin{aligned} \operatorname*{arg\,min}_{\boldsymbol{x}\in\mathbb{R}^n} u_{k+1}(\boldsymbol{x}) &\leq \operatorname*{arg\,min}_{\boldsymbol{x}\in\mathbb{R}^n} Q_{f,L_f,\boldsymbol{y}_{k+1}}(\boldsymbol{x}) \\ &= f(\boldsymbol{y}_{k+1}) - \frac{1}{2L_f}\|\nabla f(\boldsymbol{y}_{k+1})\|_2^2, \end{aligned} \tag{4.36}$$

where $\boldsymbol{x}_{k+1}$ is given by line 4 of Algorithm 1, we obtain the "general scheme of optimal method" in [16]. Note that under the assumption in (4.33), (4.36)

is satisfied not only by the upper bounds (3.8) of non-monotone ACGM in Algorithm 5, but also by the upper bounds in (3.149) of monotone ACGM in Algorithm 8. The correspondence between Nesterov's notation in [16] and ours is, for all $k \geq 0$, given by:

$$\boldsymbol{y}_k^{\mathrm{FGM}} = \boldsymbol{y}_{k+1}^{\mathrm{ACGM}}, \tag{4.37}$$

$$\alpha_k^{\mathrm{FGM}} = \frac{a_{k+1}^{\mathrm{ACGM}}}{A_{k+1}^{\mathrm{ACGM}}} = \frac{1}{t_{k+1}^{\mathrm{ACGM}}}, \tag{4.38}$$

$$\gamma_k^{\mathrm{FGM}} = \frac{\gamma_k^{\mathrm{ACGM}}}{A_k^{\mathrm{ACGM}}}. \tag{4.39}$$

The remaining state parameters are identical. Note that if $A_0^{\mathrm{ACGM}} = 0$ in (4.38), then parameter α_0^{FGM} is undefined. With assumption

$$A_0^{\mathrm{ACGM}} > 0, \tag{4.40}$$

non-monotone ACGM (Algorithm 5) is in fact identical to "constant step scheme I" in [16]. Similarly, the extrapolated form of fixed-step non-monotone ACGM (Algorithm 6) under (4.40) corresponds exactly to the "constant step scheme II" in [16]. Fixed-step border-case non-monotone ACGM (Algorithm 7), which imposes (4.40) by design, is in turn identical to the "constant step scheme III" in [16].

The FGM variant in [24] is a particular case of ACGM in Algorithm 5 (with line-search) when the objective is non-strongly convex ($\mu = 0$) and the step size search parameters are set to $r_u^{\mathrm{ACGM}} = 2$ and $r_d^{\mathrm{ACGM}} = 0.5$. The notation correspondence is as follows:

$$\boldsymbol{x}_{k+1,i}^{\mathrm{FGM}} = \boldsymbol{x}_{k+1}^{\mathrm{ACGM}}, \quad \boldsymbol{y}_{k,i}^{\mathrm{FGM}} = \boldsymbol{y}_{k+1}^{\mathrm{ACGM}}, \quad a_{k,i}^{\mathrm{FGM}} = \hat{a}^{\mathrm{ACGM}}, \quad 2^i L_f = \hat{L}^{\mathrm{ACGM}}. \tag{4.41}$$

The remaining parameters are identical.

Thus, ACGM effectively encompasses FGM [16], with its recently introduced variant [24], as well as the original FISTA [5], including its adaptive step-size variants [38,39], the monotone version MFISTA [22], the strongly convex extension FISTA-CP (including the monotone variant) [6], and the line-search extension restricted to strongly convex objectives scAPG [21]. A summary of how the above first-order methods relate to generalized ACGM is given in Table 4.7.

4.4.2 Standard WTU Worst-case Analysis

The introduction of standard WTU allows us to compare the computational convergence guarantees of ACGM with those of the state-of-the art methods applicable to the composite problem class: FGM, FISTA, FISTA-CP, scAPG, AMGS, MOS, and AA. In Subsection 4.4.1, we have seen that FGM, FISTA, FISTA-CP, and scAPG correspond to particular cases of ACGM, applicable to a subset of composite problem class and imposing particular restrictions

Table 4.7. FGM and FISTA, along with their common variants, can be considered instances of generalized ACGM with certain restrictions applied.

Algorithm	Restriction						
	Smooth objective					Fixed step size	Non-monotone
	$\mu = 0$	$\mu > 0$	$A_0 = 0$	$A_0 > 0$			
FGM [16]	yes	no	no	no	yes	yes	yes
FGM [24]	yes	yes	no	unclear	unclear	no	yes
FISTA [5]	no	yes	no	yes	no	partial	yes
MFISTA [22]	no	yes	no	yes	no	yes	no
scAPG [21]	no	no	yes	no	yes	no	yes
FISTA-CP [6]	no	no	no	no	no	yes	no

on the input parameters of ACGM. Hence, we limit our comparison to ACGM, MOS, AMGS, and AA.

Not all aforementioned methods are endowed with line search and, to be able to perform a comparison, we restrict our analysis to the composite problem class with L_f known in advance. Therefore, we study ACGM and AMGS without line-search. This setup no longer has to assume a particular parallel implementation, such as the one employing speculative execution. Therefore, the results in this section are of fundamental theoretical importance.

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

$$A_k^{\mathrm{ACGM}} = A_i^{\mathrm{ACGM}} \geq \frac{(k+1)^2}{4L_f} = \frac{(i+1)^2}{4L_f}, \tag{4.42}$$

$$A_k^{\mathrm{MOS}} = A_i^{\mathrm{MOS}} \geq \frac{k^2}{4L_f} = \frac{i^2}{4L_f}, \tag{4.43}$$

$$A_k^{\mathrm{AMGS}} = A_{\frac{i}{2}}^{\mathrm{AMGS}} \geq \frac{k^2}{2L_f} = \frac{i^2}{8L_f}, \tag{4.44}$$

$$A_k^{\mathrm{AA}} = A_{\frac{i}{2}}^{\mathrm{AA}} \geq \frac{k^2}{4L_f} = \frac{i^2}{16L_f}, \tag{4.45}$$

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

$$\frac{A_i^{\mathrm{ACGM}}}{i^2} \gtrapprox \frac{A_i^{\mathrm{MOS}}}{i^2} > \frac{A_i^{\mathrm{AMGS}}}{i^2} > \frac{A_i^{\mathrm{AA}}}{i^2}, \quad i \geq 2. \tag{4.46}$$

In the strongly convex case, let q be the inverse condition number of the objective function, defined as

$$q \stackrel{\mathrm{def}}{=} \frac{\mu}{L_f + \mu_\Psi}. \tag{4.47}$$

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 [17] either to transfer all strong convexity from f to Ψ, or to restart the algorithm at regular intervals[4]. Both enhancements have the same effect on the convergence guarantee, which can be expressed as

$$A_k^{\mathrm{AMGS}} = A_{\frac{i}{2}}^{\mathrm{AMGS}} \geq C^{\mathrm{AMGS}} \left(B^{\mathrm{AMGS}}\right)^i, \tag{4.48}$$

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

$$B^{\mathrm{AMGS}} \stackrel{\mathrm{def}}{=} \left(1 + \sqrt{\frac{\mu}{2(L_f - \mu_f)}}\right)^2 = \left(1 + \sqrt{\frac{q}{2(1-q)}}\right)^2, \tag{4.49}$$

and C^{AMGS} is a proportionality constant.

For ACGM, MOS, and AA, we have

$$A_k^{\mathrm{ACGM}} = A_i^{\mathrm{ACGM}} \geq C^{\mathrm{ACGM}} \left(B^{\mathrm{ACGM}}\right)^i, \tag{4.50}$$

$$A_k^{\mathrm{MOS}} = A_i^{\mathrm{MOS}} \geq C^{\mathrm{MOS}} \left(B^{\mathrm{MOS}}\right)^i, \tag{4.51}$$

$$A_k^{\mathrm{AA}} = A_{\frac{i}{2}}^{\mathrm{AA}} \geq C^{\mathrm{AA}} \left(B^{\mathrm{AA}}\right)^i, \tag{4.52}$$

where

$$B^{\mathrm{ACGM}} \stackrel{\mathrm{def}}{=} \frac{1}{1 - \sqrt{q}}, \tag{4.53}$$

$$B^{\mathrm{MOS}} \stackrel{\mathrm{def}}{=} \left(1 + \frac{1}{2}\sqrt{\frac{q}{1-q}}\right)^2, \tag{4.54}$$

$$B^{\mathrm{AA}} \stackrel{\mathrm{def}}{=} 1 + \frac{1}{2}\sqrt{\frac{q}{1-q}}. \tag{4.55}$$

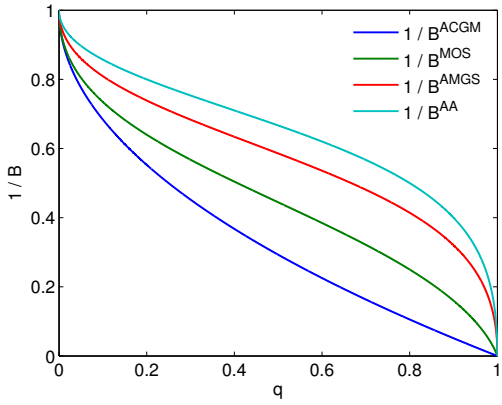
Assumption $0 < q < 1$ implies that

$$B^{\mathrm{ACGM}} > B^{\mathrm{MOS}} > B^{\mathrm{AMGS}} > B^{\mathrm{AA}}. \tag{4.56}$$

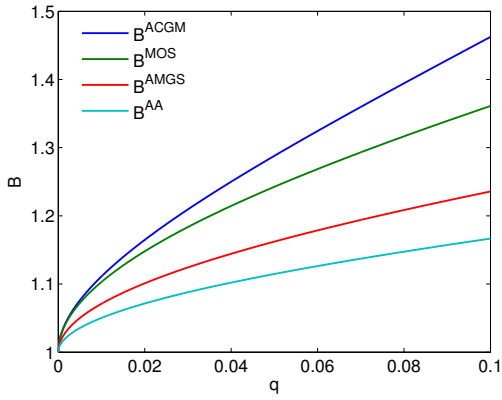
A quantitative comparison of the rates can be found in Figure 4.1. The inverse rates are compared for every possible value of q in Figure 4.1(a) whereas the rates are compared directly in Figure 4.1(b) for the range of q found in the vast majority of practical applications.

It can be clearly discerned from (4.46), (4.56), and Figure 4.1 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.

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



(a) Inverse rates as a function of q



(b) Rates for $q \leq 0.1$

Figure 4.1. Asymptotic rates of ACGM, MOS, AMGS, and AA

4.4.3 Theoretical Superiority of ACGM

Subsection 4.4.1 argues that ACGM can be considered an "umbrella" method among large-scale first-order schemes. However, ACGM is more than a generalizing framework and actually surpasses constituent FGM, FISTA, FISTA-CP, and scAPG as well as the methods in the AMGS family. For FGM and FISTA, this occurs even on the more restricted problem classes they were designed to address.

For instance, the line-search procedure of ACGM is superior to that of FISTA on non-strongly convex composite problems. FISTA's line-search suffers from two drawbacks: the parameter t_k^{FISTA} update is oblivious to the change in local curvature and the LCEs cannot decrease. Hence, if the ini-

tial guess L_0 is erroneously large, FISTA will slow down considerably (this behavior will be illustrated on a simulation example in Subsection 5.1.1). We formally express the advantages of ACGM's line-search over that of FISTA in the following proposition.

Proposition 7. *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. With judicious use of parameters r_u and r_d, the average WTU cost of an iteration of ACGM can be adjusted to equal that of FISTA (also evidenced in Subsection 5.1.1). Consequently, it is adequate to compare the convergence guarantees of the two algorithms when indexed in iterations.

Combining (3.46) and (3.48), we obtain

$$\begin{aligned} A_{k+1}^{\mathrm{ACGM}} = L_{k+1}^{\mathrm{ACGM}} \left(a_{k+1}^{\mathrm{ACGM}}\right)^2 &= \frac{1}{2L_{k+1}^{\mathrm{ACGM}}} \left(1 + \sqrt{1 + 4L_{k+1}^{\mathrm{ACGM}} A_k^{\mathrm{ACGM}}}\right) \\ &= \left(\sqrt{\frac{1}{4L_{k+1}^{\mathrm{ACGM}}}} + \sqrt{\frac{1}{4L_{k+1}^{\mathrm{ACGM}}} + A_k^{\mathrm{ACGM}}}\right)^2. \end{aligned} \tag{4.57}$$

Replacing (3.66) in non-monotone ACGM for non-strongly convex objectives in extrapolated form (Algorithm 4) with

$$t_{k+1}^{\mathrm{FISTA}} = \frac{1 + \sqrt{1 + 4(t_k^{\mathrm{FISTA}})^2}}{2}, \quad k \geq 0, \tag{4.58}$$

results in an algorithm that produces identical iterates to FISTA.

The convergence analysis of ACGM can be used to show that definition (3.57) is valid for FISTA as well. Combining (3.57) with (4.58) produces the corresponding accumulated weight for FISTA as

$$A_{k+1}^{\mathrm{FISTA}} = \left(\sqrt{\frac{1}{4L_{k+1}^{\mathrm{FISTA}}}} + \sqrt{\frac{1}{4L_{k+1}^{\mathrm{FISTA}}} + \frac{L_k^{\mathrm{FISTA}}}{L_{k+1}^{\mathrm{FISTA}}} A_k^{\mathrm{FISTA}}}\right)^2. \tag{4.59}$$

Both methods start with the same state, including $A_0^{\mathrm{ACGM}} = A_0^{\mathrm{FISTA}} = 0$. 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 an LCE increase, namely $L_k^{\mathrm{FISTA}} \leq L_{k+1}^{\mathrm{FISTA}}$, $k \geq 0$. Thus, regardless of the fluctuation in the local curvature of f, we have that

$$A_k^{\mathrm{ACGM}} \geq A_k^{\mathrm{FISTA}}, \quad k \geq 0. \tag{4.60}$$

□

The ability to dynamically and frequently adjust to the local Lipschitz constant gives ACGM an advantage over FISTA-CP, even when an accurate estimate of the Lipschitz constant is available beforehand (also argued using simulations in Subsection 5.1.2). Considering that backtracks rarely occur, the per-iteration complexity of ACGM, both in the non-strongly and strongly convex cases ($\mu \geq 0$), approaches that of FISTA and FISTA-CP (see Table 4.1) and the absolute minimum of 1 WTU per iteration. The scAPG method is an instance of border-case ACGM that features fully adaptive line-search and has a per-iteration complexity that matches the one of ACGM. However, scAPG is guaranteed to converge only when $\mu_f > 0$ and $\boldsymbol{x}_0$ is feasible, and the restriction $\gamma_0 = A_0\mu$ renders it less flexible than ACGM. This parameter choice generally has a negative impact on performance, as argued in Subsection 5.2.3.

AMGS has better iteration convergence guarantees than ACGM and is also equipped with a fully adaptive line-search procedure. However, its high per-iteration cost is almost twice that of ACGM. Consequently, ACGM surpasses AMGS in terms of computational convergence guarantees on non-strongly convex problems and actually has a better asymptotic rate on strongly convex objectives (see Subsection 4.4.2).

MOS can be considered a variant of AMGS with the per-iteration complexity of FISTA-CP. Nevertheless, MOS is slightly slower than FISTA-CP. This is proven theoretically in Subsection 4.4.2 and a practical confirmation can be found in Subsection 5.1.2. Therefore, ACGM outperforms MOS by a larger margin than FISTA-CP. AA is a variant of MOS that features an estimate sequence based acceleration heuristic which can be considered to be a form of line-search. However, AA requires that an overestimate of the Lipschitz constant be known beforehand. Unlike FISTA, where an overestimate slows down the algorithm, an underestimate in AA leads to outright divergence. Moreover, the per-iteration complexity of AA is equal to that of AMGS, that combined with an iteration convergence guarantee lower than that of FISTA-CP gives it the lowest computational convergence guarantee among the AMGS family of methods (see Subsection 4.4.2).

Table 4.8 contains a detailed feature comparison between large-scale first-order methods. The *combination* of capabilities displayed by ACGM, as outlined in Table 4.8, is unique among this class of algorithms and accounts for ACGM's superiority.

Table 4.8. Features of black-box first-order methods applicable to large-scale composite problems

Algorithm	Feature					
	Composite objective	Line-search	$\mathcal{O}(1/k^2)$ rate for $\mu = 0$	Linear rate for $\mu > 0$	$\mathcal{O}((1-\sqrt{q})^k)$ rate for $\mu > 0$	Monotone
Proximal point	yes	no	no	yes	no	yes
FGM [16]	no	no	yes	yes	yes	no
FGM [24]	no	yes	yes	no	no	no
FISTA [5]	yes	partial	yes	no	no	no
MFISTA [22]	yes	no	yes	no	no	yes
scAPG [21]	partial	yes	no	yes	yes	no
FISTA-CP [6]	yes	no	yes	yes	yes	yes
AMGS [17]	yes	yes	yes	yes	no	no
MOS [20]	yes	no	yes	yes	almost	no
AA [20]	yes	partial	yes	yes	no	no
ACGM	yes	yes	yes	yes	yes	yes

5. Simulations

5.1 Non-monotone ACGM Benchmark

In this section we test non-monotone ACGM against the state-of-the-art methods on a typical non-strongly convex inverse problem in Subsection 5.1.1 whereas in Subsection 5.1.2 we focus on a strongly convex machine learning problem. Both applications feature l_1-norm regularization [40]. 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 for which the above specialized methodologies may not apply.

5.1.1 l_1-regularized Image Deblurring

To better compare the capabilities of ACGM (Algorithm 4) to those of FISTA, we choose the very problem FISTA was introduced to solve, namely the l_1-regularized deblurring of images. For ease and accuracy of benchmarking, we have adopted the experimental setup from Section 5.1 in [5]. Here, the composite objective function is given by

$$f(\boldsymbol{x}) = \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2, \qquad \Psi(\boldsymbol{x}) = \lambda\|\boldsymbol{x}\|_1, \tag{5.1}$$

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 [41]. 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 [5] 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-

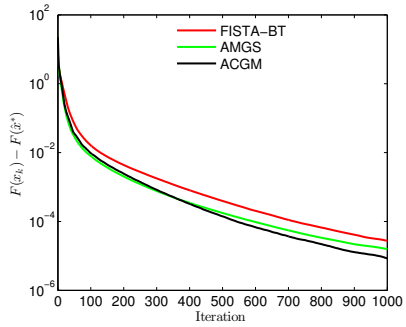
Hankel matrix (more details in [41]), which yields a value of $L_f = 2.0$. The problem is non-strongly convex with $\mu = \mu_f = \mu_\Psi = 0$. The regularization parameter λ is set to $2 \cdot 10^{-5}$ to account for the noise level of $\boldsymbol{b}$.

We have noticed that several monographs in the field (e.g., [6, 18]) do not include AMGS in their benchmarks. For completeness, we compare Algorithm 4 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 [36] to "provide good performance in many applications". We assume, as in Subsection 4.3.1, that the LCEs hover around a fixed value. Therefore, 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 (see Table 4.1). 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 $r_d^{\mathrm{ACGM}} = \sqrt{r_d^{\mathrm{AMGS}}}$.

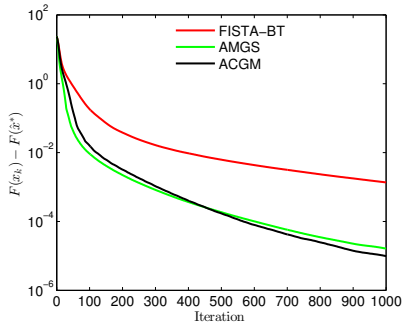
To showcase the importance of employing an algorithm with an efficient and robust line-search procedure, we have considered two scenarios: a normally underestimated initial guess $L_0 = 0.3L_f$ (Figure 5.1) and a greatly overestimated $L_0 = 10L_f$. The convergence rate is measured as the difference between 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 (Figures 5.1(a) and 5.1(b)), 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" [17]. When indexed in WTU (Figures 5.1(c) and 5.1(d)), ACGM clearly surpasses AMGS from the very beginning, because of ACGM's low per-iteration complexity.

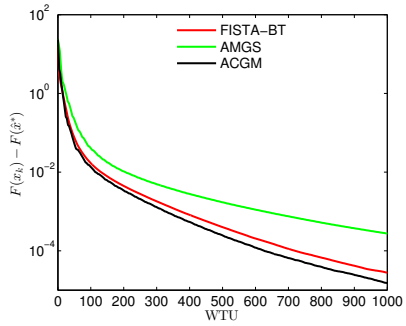
FISTA-BT lags behind in the overestimated case, regardless of the convergence measure (Figures 5.1(b) and 5.1(d)), and it is also slightly slower than ACGM in the underestimated case (Figures 5.1(a) and 5.1(c)). The disadvantage of FISTA-BT lies in the inability of its line-search procedure to decrease 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 Figures 5.1(e) and 5.1(f).



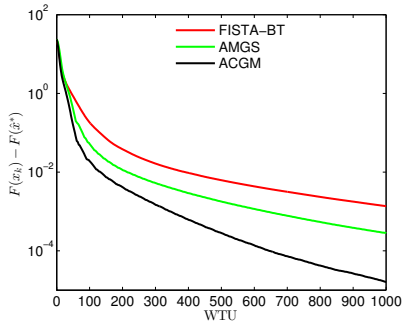
(a) Convergence rate in iterations ($L_0 = 0.3L_f$)



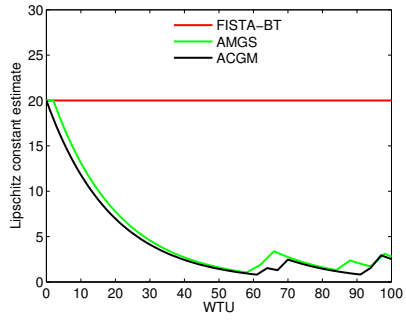
(b) Convergence rate in iterations ($L_0 = 10L_f$)



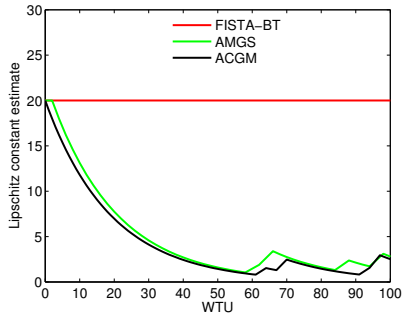
(c) Convergence rate in WTU ($L_0 = 0.3L_f$)



(d) Convergence rate in WTU ($L_0 = 10L_f$)



(e) Lipschitz constant estimates ($L_0 = 0.3L_f$)



(f) Lipschitz constant estimates ($L_0 = 10L_f$)

Figure 5.1. Convergence results on the l_1-regularized image deblurring problem ($\mu = 0$)

5.1.2 Logistic Regression with Elastic Net

As a strongly convex application, we choose a randomly generated instance of the logistic regression classification task [42], regularized with an elastic net [43]. The objective function components are, respectively, given by

$$f(\boldsymbol{x}) = -\boldsymbol{y}^T \boldsymbol{A}\boldsymbol{x} + \sum_{i=1}^{m} \log(1 + e^{\boldsymbol{a}_i^T \boldsymbol{x}}), \tag{5.2}$$

$$\Psi(\boldsymbol{x}) = \lambda_1 \|\boldsymbol{x}\|_1 + \frac{\lambda_2}{2}\|\boldsymbol{x}\|_2^2, \tag{5.3}$$

where the sparse matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ has rows $\boldsymbol{a}_i^T$, $i \in \{1, ..., m\}$, and $\boldsymbol{y} \in \mathbb{R}^m$ is the vector of classification labels. 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 matrix $\boldsymbol{A}$ is ill conditioned, which implies that the smooth part f is *not* strongly convex ($\mu_f = 0$).

The elastic net regularizer Ψ has parameters λ_1 and λ_2. Therefore, the strong convexity of the objective is $\mu = \mu_\Psi = \lambda_2$. Elastic net regularization is specified by the user [43] and we assume that optimization algorithms can access μ_Ψ.

In this simulation, the problem size is $m = n = 10^4$. The matrix $\boldsymbol{A}$ has 10% of 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

$$\mathbb{P}(\boldsymbol{Y}_i = 1) = \frac{1}{1 + e^{\langle \boldsymbol{a}_i^T, \boldsymbol{x}\rangle}}, \quad i \in \{1, ..., m\}. \tag{5.4}$$

The elastic net parameters are $\lambda_1 = 1$ and $\lambda_2 = 10^{-3}L_\sigma$.

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 Subsection 4.4.2 for proof), or invalidate it altogether, like adaptive restart in FISTA [44] or in AA [20].

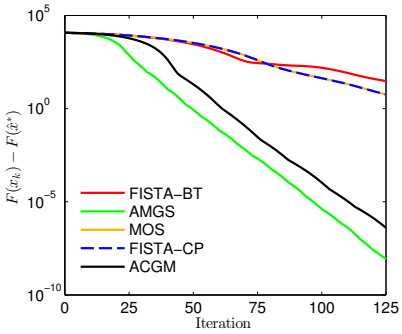
The starting point $\boldsymbol{x}_0$, the same for all algorithms tested, has entries randomly sampled as i.i.d. from $\mathcal{N}(0, 1)$. For ACGM, we set $A_0 = 0$. In this case, the value of γ_0 has no influence. The effectiveness of this initial parameter selection will be argued in Subsection 5.2.3. For the same reasons as outlined in Subsection 5.1.1, 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 $\hat{x}^*$ 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 methods in the accurate $L_f = L_\sigma$ case as well as the overestimated $L_f = 5L_\sigma$ case (Figure 5.2).

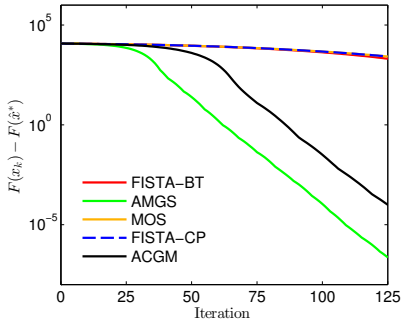
When indexing in iterations, AMGS converges the fastest (Figures 5.2(a) and 5.2(b)). However, AMGS has the same asymptotic rate in iterations as ACGM, despite AMGS performing around twice the number of proximal gradient steps per iteration. While proximal gradient steps (incurring 1 WTU each) in AMGS improve the Lipschitz constant estimate (Figures 5.2(e) and 5.2(f)), they do not appear to be used efficiently in advancing the algorithm since AMGS is inferior to ACGM and FISTA-CP in terms of WTU usage (Figures 5.2(c) and 5.2(d)). Note that FISTA-CP and MOS display nearly identical convergence behaviors (Figures 5.2(a), 5.2(b), 5.2(c), and 5.2(d)), as theoretically argued in Subsection 4.4.2.

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 (Figures 5.2(a) and 5.2(c)). The reason is that the Lipschitz estimates of ACGM are several times smaller than the global value L_f (Figure 5.2(e)). 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[1]. The benefit of being able to decrease the LCE at runtime is predictably more evident in the inaccurate case (Figure 5.2(f)). Even though AMGS also features a fully adaptive line-search procedure, estimates produced by the AMGS's "damped relaxation condition" are considerably higher than those of ACGM, further contributing for the ACGM's superior convergence behavior in WTU (Figures 5.2(c) and 5.2(d)).

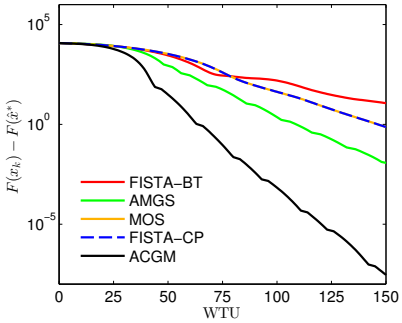
[1]We forward the reader to [6] for a more detailed comparison between FISTA and FISTA-CP.



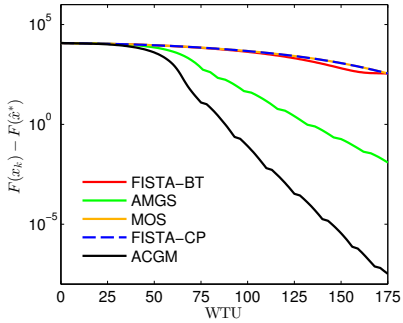
(a) Convergence rate in iterations ($L_f = L_\sigma$)



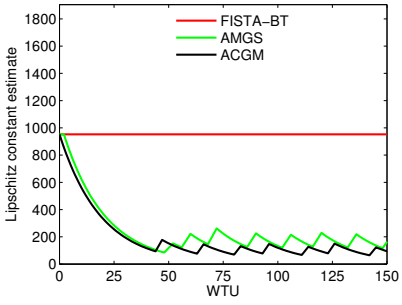
(b) Convergence rate in iterations ($L_f = 5L_\sigma$)



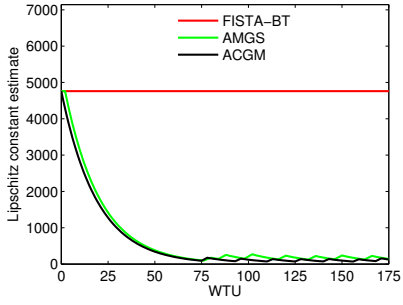
(c) Convergence rate in WTU ($L_f = L_\sigma$)



(d) Convergence rate in WTU ($L_f = 5L_\sigma$)



(e) Lipschitz constant estimates ($L_f = L_\sigma$)



(f) Lipschitz constant estimates ($L_f = 5L_\sigma$)

Figure 5.2. Convergence results on logistic regression with elastic net ($\mu = 10^{-3}L_\sigma$)

5.2 Monotone ACGM Benchmark

5.2.1 Benchmark Setup

Having confirmed the superiority of ACGM on two simple examples, we continue the tests using a larger selection of parameters and variants of ACGM, including the monotone ones. We now include in our benchmark non-monotone ACGM (denoted as plain ACGM), monotone ACGM (MACGM), and, for strongly-convex problems, border-case non-monotone ACGM (BACGM) as well as border-case monotone ACGM (BMACGM). We compare against FISTA-CP, monotone FISTA-CP (MFISTA-CP), AMGS, and FISTA with backtracking line-search (FISTA-BT). BACGM produces identical iterates to scAPG, so we do not mention scAPG explicitly in our simulations.

In this extended survey, we have selected as test cases five synthetic instances of composite problems in the areas of statistics, inverse problems, and machine learning. Three are non-strongly convex: the least absolute shrinkage and selection operator (LASSO) [40], non-negative least squares (NNLS), and l_1-regularized logistic regression (L1LR). The other two are strongly-convex: ridge regression (RR) and elastic net (EN) [43]. Table 5.1 lists the oracle functions of all above mentioned problems. Here, the sum

Table 5.1. Oracle functions of the five test problems

Problem	$f(\boldsymbol{x})$	$\Psi(\boldsymbol{x})$	$\nabla f(\boldsymbol{x})$	$prox_{\tau\Psi}(\boldsymbol{x})$
LASSO	$\frac{1}{2}\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_2^2$	$\lambda_1\|\boldsymbol{x}\|_1$	$\boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b})$	$\mathcal{T}_{\tau\lambda_1}(\boldsymbol{x})$
NNLS	$\frac{1}{2}\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_2^2$	$\sigma_{\mathbb{R}^n_+}(\boldsymbol{x})$	$\boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b})$	$(\boldsymbol{x})_+$
L1LR	$\mathcal{I}(\boldsymbol{A}\boldsymbol{x})-\boldsymbol{y}^T\boldsymbol{A}\boldsymbol{x}$	$\lambda_1\|\boldsymbol{x}\|_1$	$\boldsymbol{A}^T(\mathcal{L}(\boldsymbol{A}\boldsymbol{x})-\boldsymbol{y})$	$\mathcal{T}_{\tau\lambda_1}(\boldsymbol{x})$
RR	$\frac{1}{2}\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_2^2$	$\frac{\lambda_2}{2}\|\boldsymbol{x}\|_2^2$	$\boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b})$	$\frac{1}{1+\tau\lambda_2}\boldsymbol{x}$
EN	$\frac{1}{2}\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_2^2$	$\lambda_1\|\boldsymbol{x}\|_1+\frac{\lambda_2}{2}\|\boldsymbol{x}\|_2^2$	$\boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b})$	$\frac{1}{1+\tau\lambda_2}\mathcal{T}_{\tau\lambda_1}(\boldsymbol{x})$

softplus function $\mathcal{I}(\boldsymbol{x})$, the element-wise logistic function $\mathcal{L}(\boldsymbol{x})$, and the shrinkage operator $\mathcal{T}_\tau(\boldsymbol{x})$ are, respectively, given by

$$\mathcal{I}(\boldsymbol{x}) = \sum_{i=1}^{m} \log\left(1+e^{\boldsymbol{x}_i}\right), \quad i \in \{1,...,m\}, \tag{5.5}$$

$$\mathcal{L}(\boldsymbol{x})_i = \frac{1}{1+e^{-\boldsymbol{x}_i}}, \quad i \in \{1,...,m\}, \tag{5.6}$$

$$\mathcal{T}_\tau(\boldsymbol{x})_j = (|\boldsymbol{x}_j|-\tau)_+ \operatorname{sgn}(\boldsymbol{x}_j), \quad j \in \{1,...,n\}. \tag{5.7}$$

To attain the best convergence guarantees for AMGS, Nesterov suggests in [17] that all known global strong convexity be transfered to the simple function Ψ. When line-search is enabled, generalized ACGM also benefits slightly from this arrangement when $r_u > 1$ because the LCEs are smaller and therefore tighter. Without line-search, the convergence guarantees of

generalized ACGM do not change as a result of strong convexity transfer, in either direction. Thus, for fair comparison, we have incorporated in Ψ the strongly-convex quadratic regularization term for the RR and EN problems. In the following, we describe in detail each of the five problem instances. All random variables are independent and identically distributed, unless stated otherwise.

LASSO. Real-valued matrix $\boldsymbol{A}$ is of size $m = 500 \times n = 500$, with entries drawn from $\mathcal{N}(0,1)$. Vector $\boldsymbol{b} \in \mathbb{R}^m$ has entries sampled from $\mathcal{N}(0,9)$. Regularization parameter λ_1 is 4. The starting point $\boldsymbol{x}_0 \in \mathbb{R}^n$ has entries drawn from $\mathcal{N}(0,1)$.

NNLS. Sparse $m = 1000 \times n = 10000$ matrix $\boldsymbol{A}$ has approximately 10% of entries, at random locations, non-zero. The non-zero entries are drawn from $\mathcal{N}(0,1)$ after which each column $j \in \{1,...,n\}$ is scaled independently to have an l_2 norm of 1. Starting point $\boldsymbol{x}_0$ has 10 entries at random locations all equal to 4 and the remainder zero. Vector $\boldsymbol{b}$ is obtained from $\boldsymbol{b} = \boldsymbol{A}\boldsymbol{x}_0 + \boldsymbol{z}$, where $\boldsymbol{z}$ is standard Gaussian noise.

L1LR. Matrix $\boldsymbol{A}$ has $m = 200 \times n = 1000$ entries sampled from $\mathcal{N}(0,1)$, $\boldsymbol{x}_0$ has exactly 10 non-zero entries at random locations, each entry value drawn from $\mathcal{N}(0,225)$, and $\lambda_1 = 5$. Labels $\boldsymbol{y}_i \in \{0,1\}, i \in \{1,...,m\}$, are selected with probability $\mathbb{P}(\boldsymbol{Y}_i = 1) = \mathcal{L}(\boldsymbol{A}\boldsymbol{x})_i$.

RR. Dimensions are $m = 500 \times n = 500$. The entries of matrix $\boldsymbol{A}$, vector $\boldsymbol{b}$, and starting point $\boldsymbol{x}_0$ are drawn from $\mathcal{N}(0,1)$, $\mathcal{N}(0,25)$, and $\mathcal{N}(0,1)$, respectively. Regularizer λ_2 is given by $10^{-3}(\sigma_{max}(\boldsymbol{A}))^2$, where $\sigma_{max}(\boldsymbol{A})$ is the largest singular value of $\boldsymbol{A}$.

EN. Matrix $\boldsymbol{A}$ has $m = 1000 \times n = 500$ entries sampled from $\mathcal{N}(0,1)$. Starting point $\boldsymbol{x}_0$ has 20 non-zero entries at random locations, each entry value drawn from $\mathcal{N}(0,1)$. Regularization parameter λ_1 is obtained according to [45] as $\lambda_1 = 1.5\sqrt{2\log(n)}$ and λ_2 is the same as in RR, namely $\lambda_2 = 10^{-3}(\sigma_{max}(\boldsymbol{A}))^2$.

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

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

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

The line-search parameters were selected according to the recommendation given in [36]. For AMGS and FISTA-BT we have $r_u^{\mathrm{AMGS}} = r_u^{\mathrm{FISTA}} = 2.0$

and $r_d^{\mathrm{AMGS}} = 0.9$. The variants of generalized ACGM and AMGS are the only methods included in the benchmark that are equipped with fully adaptive line-search. We have decided to select r_d^{ACGM} to ensure that ACGM and AMGS have the same overhead. We formally define the line-search overhead of method $\mathcal{M}$, denoted by $\Omega^{\mathcal{M}}$, as the average computational cost attributable to backtracks per WTU of advancement. Assuming that the LCEs hover around a fixed value (see also Subsection 5.1.1), we thus have that

$$\Omega^{\mathrm{AMGS}} = -\frac{(2t_g + t_p)\log(r_d^{\mathrm{AMGS}})}{2(t_g + t_p)\log(r_u^{\mathrm{AMGS}})}, \quad \Omega^{\mathrm{ACGM}} = -\frac{(t_f + t_g + t_p)\log(r_d^{\mathrm{ACGM}})}{(t_g + t_p)\log(r_u^{\mathrm{ACGM}})}. \tag{5.8}$$

From (5.8) we have that $r_d^{\mathrm{ACGM}} = (r_d^{\mathrm{AMGS}})^{\frac{2}{3}}$, with no difference for border-case or monotone variants.

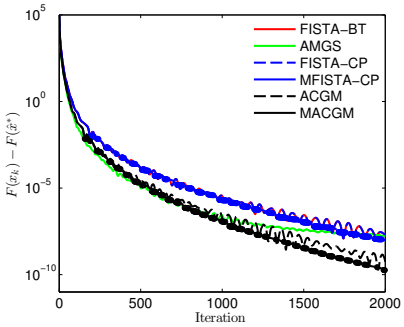
For measuring ISDs, we have computed beforehand an optimal point estimate $\hat{\boldsymbol{x}}^*$ for each problem instance. Each $\hat{\boldsymbol{x}}^*$ was obtained as the main iterate after running MACGM for 5000 iterations with parameters $A_0 = 0$, $\gamma_0 = 1$, $L_0 = L_f$, and aggressive search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

5.2.2 Non-strongly Convex Problems

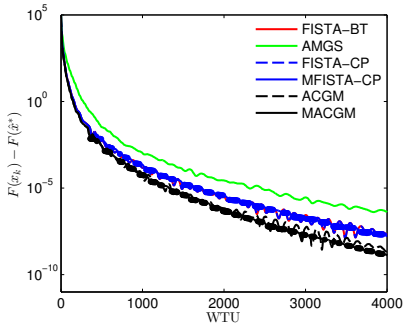
The convergence results for LASSO, NNLS, and L1LR are shown in Figure 5.3. The LCE variation during the first 200 WTU is shown in Figures 5.4(a) and 5.4(c) for LASSO and L1LR, respectively. For NNLS, floating point precision was exhausted after 100 WTU and the LCE variation was only studied up to this point (Figure 5.4(b)). In addition, the average LCEs are listed in Table 5.2.

Both variants of ACGM outperform in iterations and especially in WTU the competing methods in each of these problem instances (Figure 5.3). Even though for LASSO and NNLS, the iteration convergence rate of AMGS is slightly better in the beginning (Figures 5.3(a) and 5.3(c)), AMGS lags behind afterwards and, when measured in terms of computational convergence rate, has the poorest performance among the methods tested (Figures 5.3(b) and 5.3(d)). In all the non-strongly convex problems, when $L_0 = L_f$, the backtrack condition of FISTA-BT is never triggered and FISTA-BT produces the same iterates as FISTA-CP (Figure 5.3).

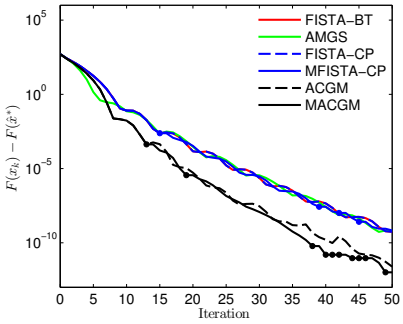
The overall superiority of ACGM and MACGM can be attributed to the effectiveness of line-search. Interestingly, ACGM manages to surpass FISTA-CP and MFISTA-CP even when the latter are supplied with the exact value of the global Lipschitz constant. This is because ACGM is able to accurately estimate the local curvature, which is often below L_f. For the L1LR problem, where the smooth part f is not the square of a linear function, the local curvature is substantially lower than the global Lipschitz constant with LCEs hovering around one fifth of L_f (Figure 5.4(c)). AMGS is not able to estimate local curvature as accurately as ACGM



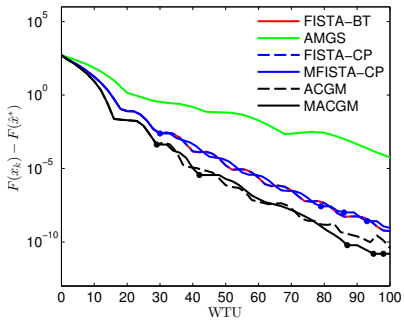
(a) Iteration convergence rates on LASSO



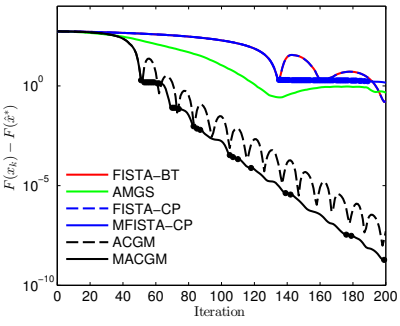
(b) Computational convergence rates on LASSO



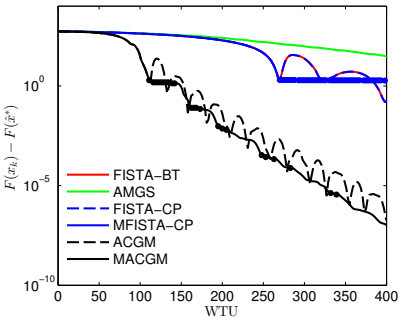
(c) Iteration convergence rates on NNLS



(d) Computational convergence rates on NNLS

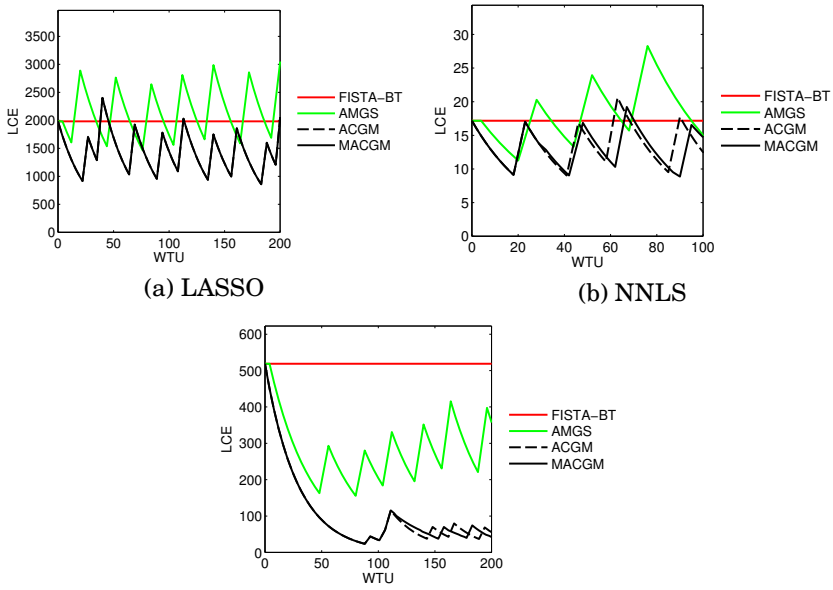


(e) Iteration convergence rates on L1LR



(f) Computational convergence rates on L1LR

Figure 5.3. Convergence results of FISTA with backtracking (FISTA-BT), AMGS, FISTA-CP, monotone FISTA-CP (MFISTA-CP), non-monotone ACGM and monotone ACGM (MACGM) on the LASSO, NNLS, and L1LR non-strongly convex problems. Dots mark iterations *preceding* overshoots. At these iterations, the convergence behavior changes.



(a) LASSO

(b) NNLS

(c) L1LR

Figure 5.4. Line-search method LCE variation on LASSO, NNLS, and L1LR

Table 5.2. Average LCEs of line-search methods on LASSO, NNLS, and L1LR

Problem	L_f	Iterations	FISTA-BT	AMGS	ACGM	MACGM
LASSO	1981.98	2000	1981.98	2202.66	1385.85	1303.70
NNLS	17.17	50	17.17	19.86	14.35	13.54
L1LR	518.79	200	518.79	246.56	80.76	79.12

due to AMGS's reliance on a "damped relaxation condition" line-search stopping criterion. For LASSO and NNLS, the average LCE of AMGS is actually above L_f (Table 5.2). ACGM has an average LCE that is roughly two thirds that of AMGS on these problems whereas for L1LR the average is more than three times lower than AMGS. The difference between the LCE averages of ACGM and MACGM is negligible.

Indeed, monotonicity, as predicted, does not alter the overall iteration convergence rate and has a stabilizing effect. MACGM overshoots do have a negative but limited impact on the computational convergence rate. We have noticed in our simulations that overshoots occur less often[2] for larger problems, such as the tested instance of NNLS.

[2]Our computing system used double precision floating point internally so we consider only the first 12 digits of image space accuracy to be reliable measurements.

5.2.3 Strongly Convex Problems

The convergence results for RR and EN are shown in Figures 5.5(a), 5.5(b), 5.5(c), and 5.5(d). The LCE variation is shown in Figure 5.6 with the LCE averages listed in Table 5.3.

Strongly convex problems have a unique optimum point and accelerated first-order schemes are guaranteed to find an accurate estimate of it in domain space (see [16] for a detailed analysis). Since our procedure for obtaining $\hat{\boldsymbol{x}}^*$ ensures that $\hat{\boldsymbol{x}}^* \simeq \boldsymbol{x}^*$, we have that

$$A_0(F(\boldsymbol{x}_0) - F(\hat{\boldsymbol{x}}^*)) + \frac{\gamma_0}{2}\|\boldsymbol{x}_0 - \hat{\boldsymbol{x}}^*\|_2^2 \simeq \Delta_0. \tag{5.9}$$

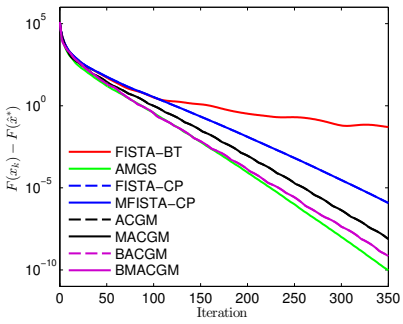
Thus, we can display accurate estimates U_k of ISDUBs in (2.5), defined as

$$U_k \stackrel{\text{def}}{=} \frac{\Delta_0}{A_k}, \quad k \geq 1, \tag{5.10}$$

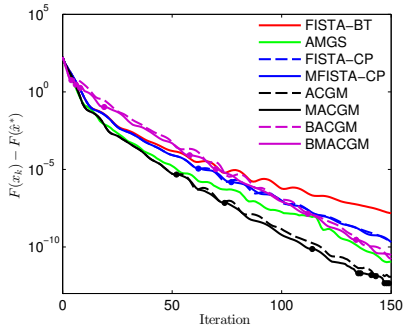
for methods that maintain convergence guarantees at runtime. These are shown in Figures 5.5(e) and 5.5(f) as upper bounds indexed in WTU.

For the RR problem, all methods except FISTA-BT exhibit a smooth linear convergence rate (Figures 5.5(a) and 5.5(c)). In iterations, AMGS converges the fastest. However, in terms of WTU usage, it is the least effective of the methods designed to deal with strongly convex objectives (Figure 5.5(c)). The reasons are the high cost of its iterations, its low asymptotic rate compared to ACGM and FISTA-CP (see Subsection 4.4.2), and the stringency of its damped relaxation criterion that results in higher LCEs (on average) than ACGM (Figure 5.6(a) and Table 5.3). The computational convergence rate of BACGM is the best, followed by ACGM, FISTA-CP, and AMGS. This does not, however, correspond to the upper bounds (Figure 5.5(e)). While BACGM produces the largest accumulated weights A_k, the high value of the ISD term in Δ_0 causes BACGM to have poorer upper bounds than ACGM, except for the first iterations. In fact, the effectiveness of BACGM on this problem is exceptional, partly due to the regularity of the composite gradients. This regularity also ensures monotonicity of BACGM, ACGM, and FISTA-CP. FISTA-BT is effective during the first 200 WTU, after which it lags behind all other methods. After 500 WTU, FISTA-BT is even slower than AMGS, despite its lower line-search overhead and advantageous parameter choice $L_0 = L_f$.

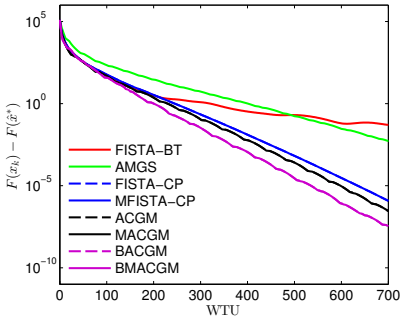
On the less regular EN problem, ACGM leads all other methods in terms of both iteration and computational convergence rates (Figures 5.5(b) and 5.5(d)). The advantage of ACGM, especially over BACGM, is accurately reflected in the upper bounds (Figure 5.5(f)). However, convergence is much faster than the upper bounds would imply. Even FISTA-BT has a competitive rate, due to the small number of iterations (150) needed for high accuracy results. The ineffectiveness of AMGS on this problem is mostly due to its high LCEs (Figure 5.6(b) and Table 5.3). All variants



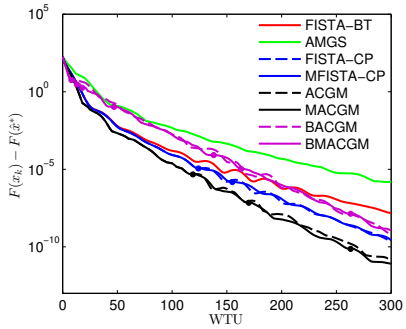
(a) Iteration convergence rates on RR



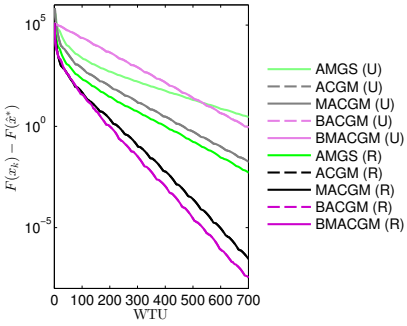
(b) Iteration convergence rates on EN



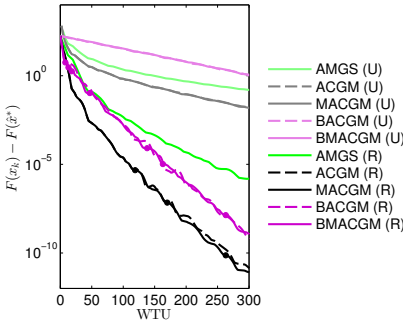
(c) Computational convergence rates on RR



(d) Computational convergence rates on EN

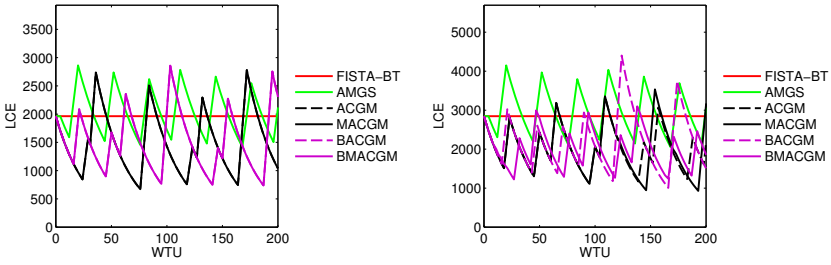


(e) Computational convergence rates (R) and upper bounds (U) on RR



(f) Computational convergence rates (R) and upper bounds (U) on EN

Figure 5.5. Convergence results of FISTA with backtracking (FISTA-BT), AMGS, FISTA-CP, monotone FISTA-CP (MFISTA-CP), non-monotone ACGM, monotone ACGM (MACGM), border-case non-monotone ACGM (BACGM), and border-case monotone ACGM (BMACGM) on the RR and EN strongly-convex problems. Dots mark iterations preceding overshoots.



(a) RR (b) EN

Figure 5.6. Line-search method LCE variation on RR and EN

Table 5.3. Average LCEs of line-search methods on RR and EN. RR has $L_f = 1963.66$ and for EN $L_f = 2846.02$. All algorithms were run on RR for $K = 350$ iterations and on EN for $K = 150$ iterations.

Algorithm	Problem	
	RR	EN
FISTA-BT	1963.66	2846.02
AMGS	2022.73	3023.47
ACGM	1473.88	2056.68
MACGM	1473.88	2003.09
BACGM	1471.16	2093.56
BMACGM	1471.16	1998.12

of ACGM have comparable average LCEs. Here as well, monotonicity has a stabilizing effect and does not have a significant impact on the computational convergence rate.

We note that for both the RR and EN problems, regardless of the actual performance of BACGM, the convergence guarantees of BACGM are poorer than those of ACGM with $A_0 = 0$. This discrepancy in guarantees is supported theoretically because, in the most common applications, the ISD term in Δ_0 is large compared to the DST. This extends to the fixed-step setup and challenges the notion found in the literature (e.g., [46]) that for strongly-convex functions, FGM and FISTA-CP are momentum methods that take the form of the "constant step scheme III" in [16]. In fact, the border-case form may constitute the poorest choice of parameters A_0 and γ_0 in many applications. Indeed, the worst-case results in Theorem 7 favor $A_0 = 0$.

6. Ultrasound Image Reconstruction

The reconstruction of ultrasound images is a difficult open problem, due to the high degree of noise, the lack of clearly defined features, and the high dynamic range of ultrasound images. In this chapter, we shall utilize the low resource usage, applicability, adaptability, and state-of-the art convergence guarantees of ACGM to develop an accurate ultrasound image reconstruction methodology that is tractable even for large images.

6.1 Background

Ultrasound imaging is an efficient, cost effective, and safe medical imaging modality. It is widely used for various clinical applications and is especially well suited for the diagnosis of soft tissue pathologies. These advantages are however mitigated by the relative low image quality (in terms of signal-to-noise ratio), low contrast, and poor spatial resolution. The main factors affecting the quality of ultrasound images are the finite bandwidth and aperture of the imaging transducer as well as the physical phenomena (e.g., diffraction and attenuation) related to the propagation of sound waves in human tissues. Consequently, a rich body of scientific literature addresses ultrasound image reconstruction, i.e., the estimation of the tissue reflectivity function (TRF) from ultrasound radio-frequency (RF) images. Generally, existing approaches turn the TRF estimation into a deconvolution problem, by considering, under the first order Born approximation, that the formation of ultrasound images follows a 2D convolution model between the TRF and a system point-spread function (PSF). The PSF can be either estimated in a pre-processing step (see, e.g., [47–52]) or jointly estimated with the TRF, an approach known as blind deconvolution (see, e.g., [53–56]). Mainly for computational reasons, most of the existing ultrasound image restoration methods consider a spatially invariant PSF model and circulant boundary conditions. However, regardless of the acquisition setup, stationary convolution cannot accurately model the formation of ultrasound images.

6.1.1 Pulse-echo Ultrasound

Pulse-echo emission of focused waves still remains the most widely used acquisition scheme in ultrasound imaging. The active elements of the ultrasound probe sequentially transmit an excitation, unique for each element, such that the combined beam is narrowly focused at a certain location in the tissue. Scatterers in the tissue reflect the ultrasound waves back to the probe. The active elements in the probe are piezoelectric and also act as receivers. The received signals, with one raw channel data stream corresponding to each active element, are combined in a process known as beamforming to obtain one radio-frequency (RF) signal. These focused beams are transmitted sequentially. For each transmission centered at a lateral position, the raw data is used to beamform one RF signal. The juxtaposition of the RF signals yields the RF image.

Each RF signal passes through a band-pass filter. The Hilbert transform is applied and the magnitude of the result constitutes the detected envelope. The log-compressed envelopes of the RF channel signals are juxtaposed to form the ultrasound image that is displayed to the user. The process of obtaining the final ultrasound image from the RF image is well defined and will not be developed further in this work. Instead, our goal is to recover the TRF from the RF image.

Whereas the RF image formation process is poorly modeled by traditional 2D convolution, a spatially varying kernel convolution model can be very accurate [57, 58]. Given the repeatability of the imaging process in the lateral direction, the lateral variation of the kernels is negligible. However, despite dynamic focusing in reception and time gain compensation, the kernels become wider as we move away from the focal depth, thus degrading the spatial resolution.

6.1.2 Previous Work

The kernel variation has been accounted for by assuming local regions of kernel invariance and performing deconvolution block-wise (e.g., [59]). Very recently, ultrasound imaging convolution models with continuously varying kernels were proposed in [57] and [58]. However, [57] makes the overly restrictive assumption that the spatially varying kernel is obtained from a constant reference kernel modulated by the exponential of a fixed discrete generator scaled by the varying kernel center image coordinates. Therefore, it does not take into account the depth-dependent spatial resolution degradation that occurs in pulse-echo ultrasound. On the other hand, the deconvolution proposed in [58] has an iteration complexity proportional to the cube of the number of pixels in the image, limiting its applicability to very small images.

6.2 Notation

In this chapter, images (ultrasound images and TRFs) are vectorized in column-major order but referenced in 2D form. For instance, image $\boldsymbol{v} \in \mathbb{R}^{m_v n_v}$ corresponds to an $m_v \times n_v$ 2D image and has the pixel value at coordinates (i, j) given by $v_{m_v(j-1)+i}$. However, for clarity of exposition, we denote it as the 2D object $\boldsymbol{v} \in \mathbb{R}^{m_v \times n_v}$, with the pixel value at location (i, j) given by $\boldsymbol{v}_{i,j}$. Bold marks this artificial indexing. Similarly, linear operators are matrices but referred to as 4D tensors, e.g., $\boldsymbol{O} : \mathbb{R}^{m_v \times n_v} \to \mathbb{R}^{m_w \times n_w}$ denotes $\boldsymbol{O} \in \mathbb{R}^{m_v n_v \times m_w n_w}$.

In the sequel, we define several classes of linear operators that constitute the mathematical building blocks of our proposed model and analysis. Note that these are more general than normal linear operators because their dimensions not only depend on those of their parameters but also of their arguments. These arguments include *kernel* $\boldsymbol{k} \in \mathbb{R}^{m_k \times n_k}$ and *image* $\boldsymbol{a} \in \mathbb{R}^{m_a \times n_a}$, where $m_k, n_k, m_a, n_a \geq 1$.

Let the rotation operator $\boldsymbol{\mathcal{R}}(\boldsymbol{k})$ be given by

$$(\boldsymbol{\mathcal{R}}(\boldsymbol{k}))_{i,j} \stackrel{\text{def}}{=} \boldsymbol{k}_{m_k-i+1,n_k-j+1}, \quad i \in \{1, ..., m_k\}, \quad j \in \{1, ..., n_k\}. \tag{6.1}$$

To simplify notation, for all indices $1 \leq a \leq b \leq c$, we denote the exception index set $\mathcal{I}(a, b, c)$ as

$$\mathcal{I}(a, b, c) \stackrel{\text{def}}{=} \{1, ..., c\} \setminus \{a, ..., b\}. \tag{6.2}$$

The window and zero padding operators are, respectively, defined as

$$\boldsymbol{\mathcal{W}}(i_1, i_2, j_1, j_2, m_a, n_a)\boldsymbol{a} \stackrel{\text{def}}{=} \boldsymbol{a}_{i+i_1,j+j_1}, \ i \in \{1, ..., i_2 - i_1\}, \ j \in \{1, ..., j_2 - j_1\}, \tag{6.3}$$

$$\boldsymbol{\mathcal{Z}}(i_1, i_2, j_1, j_2, m_a, n_a)\boldsymbol{a} \stackrel{\text{def}}{=} \begin{cases} \boldsymbol{a}_{i-i_1,j-j_1}, & i_1 \leq i \leq i_2, \ j_1 \leq j \leq j_2, \\ 0, & i \in \mathcal{I}(i_1, i_2, m_a), \ j \in \mathcal{I}(j_1, j_2, n_a). \end{cases} \tag{6.4}$$

It trivially follows that the two operators are mutually adjoint, namely

$$\boldsymbol{\mathcal{W}}(i_1, i_2, j_1, j_2, m_a, n_a)^T = \boldsymbol{\mathcal{Z}}(i_1, i_2, j_1, j_2, m_a, n_a), \tag{6.5}$$

where $(*)^T$ denotes the adjoint of a linear operator, which corresponds to the transpose of its matrix form. Their effect on a test image is shown in Figure 6.1.

Note that none of the operators introduced up until now involve any computation in practice.

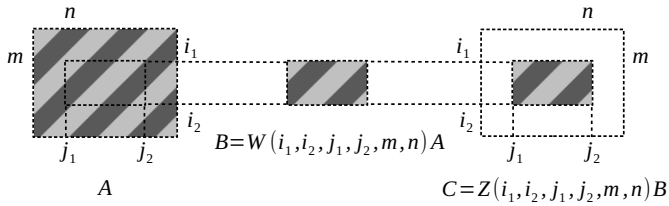


Figure 6.1. Applying the full-width window operator, followed by a full-width zero-padding operator on a test image $\boldsymbol{a}$.

6.3 Discrete Convolution

6.3.1 Definitions

To define spatially varying convolution, it is necessary to first define spatially invariant convolution for images or image patches of arbitrary size.

Circular convolution
Existing ultrasound formation forward models assume that the RF image is formed from the discrete circular convolution of the TRF with a system PSF. Circular (periodic) convolution approximates the physical processes involved in the formation of the central part of the image while being very computationally efficient [60]. It only applies to arguments of equal size. When $\boldsymbol{k}$ and $\boldsymbol{a}$ are not the same size, they can only be circularly convolved by appropriately padding them with zeros. When dealing with circular convolution, we only consider arguments $\bar{\boldsymbol{k}}$ and $\bar{\boldsymbol{a}}$ of equal size m by n. Circular convolution $\circledast$ can be defined using the discrete Fourier transform (DFT) using the convolution theorem [61] as

$$\bar{\boldsymbol{k}} \circledast \bar{\boldsymbol{a}} \stackrel{\text{def}}{=} \boldsymbol{F}^H((\boldsymbol{F}\bar{\boldsymbol{k}}) \odot (\boldsymbol{F}\bar{\boldsymbol{a}})), \tag{6.6}$$

where $\boldsymbol{F}$ is the DFT operator.

Circular convolution in (6.6) can be written explicitly [62], without employing the DFT, as

$$(\bar{\boldsymbol{k}} \circledast \bar{\boldsymbol{a}})_{i,j} = \sum_{p=1}^{m}\sum_{q=1}^{n} \bar{\boldsymbol{k}}_{p,q}\bar{\boldsymbol{a}}_{i\ominus_m p, j\ominus_n q}, \quad i \in \{1,...,m\}, \quad j \in \{1,...,n\}, \tag{6.7}$$

where the circular sum $\oplus$ and difference $\ominus$ are, respectively, defined as

$$a \oplus_c b \stackrel{\text{def}}{=} ((a+b-2) \bmod c) + 1, \tag{6.8}$$

$$a \ominus_c b \stackrel{\text{def}}{=} ((a-b) \bmod c) + 1, \tag{6.9}$$

for all integers $c \geq 1$ and $a, b \in \{1,...,c\}$.

The circular convolution linear operator $\boldsymbol{\mathcal{C}}(\bar{\boldsymbol{k}})$, parametrized by kernel $\bar{\boldsymbol{k}}$ is defined as

$$\boldsymbol{\mathcal{C}}(\bar{\boldsymbol{k}})\bar{\boldsymbol{a}} \stackrel{\text{def}}{=} \bar{\boldsymbol{k}} \circledast \bar{\boldsymbol{a}}. \tag{6.10}$$

Valid and full convolution

Whereas circular convolution possesses remarkable mathematical properties, the zero padding of the arguments and its circulant boundary are entirely artificial. Full convolution very accurately models the ultrasound image formation process from isolated scatterers [58, 63].

We define discrete full convolution $*_1$ between kernel $\boldsymbol{k} \in \mathbb{R}^{m_k \times n_k}$ and image $\boldsymbol{a} \in \mathbb{R}^{m_a \times n_a}$ of size $1 \leq m_k \leq m_a$ and $1 \leq n_k \leq n_a$ as

$$\begin{aligned} (\boldsymbol{k} *_1 \boldsymbol{a})_{i,j} &\stackrel{\text{def}}{=} \sum_{p=p_i}^{\bar{p}_i} \sum_{q=q_j}^{\bar{q}_j} \boldsymbol{k}_{p,q} \boldsymbol{a}_{i-p+1,j-q+1}, \\ i \in \{1, ..., m_a + n_k - 1\},\ j &\in \{1, ..., n_a + n_k - 1\}, \\ p_i = \max\{1, i - m_a + 1\}, &\quad \bar{p}_i = \min\{i, m_k\}, \\ q_j = \max\{1, j - n_a + 1\}, &\quad \bar{q}_j = \min\{j, n_k\}. \end{aligned} \tag{6.11}$$

Its corresponding linear operator $\boldsymbol{\mathcal{C}}_1(\boldsymbol{k})$, parametrized by kernel $\boldsymbol{k}$, is defined as

$$\boldsymbol{\mathcal{C}}_1(\boldsymbol{k})\boldsymbol{a} \stackrel{\text{def}}{=} \boldsymbol{k} *_1 \boldsymbol{a}, \tag{6.12}$$

for all $\boldsymbol{a} \in \mathbb{R}^{m_a \times n_a}$

Valid convolution is the subset of full convolution where every output pixel is expressed using the entire kernel $\boldsymbol{k}$. This type of convolution is particularly useful when the reconstructed TRF needs to be as large as the RF image [64]. Valid convolution $*_2$ is written explicitly as

$$\begin{aligned} (\boldsymbol{k} *_2 \boldsymbol{a})_{i,j} &\stackrel{\text{def}}{=} \sum_{p=1}^{m_k} \sum_{q=1}^{n_k} \boldsymbol{k}_{p,q} \boldsymbol{a}_{i-p+m_k,j-q+n_k}, \\ i \in \{1, ..., m_a - m_k + 1\},\ j &\in \{1, ..., n_a - n_k + 1\}, \end{aligned} \tag{6.13}$$

with its linear operator $\boldsymbol{\mathcal{C}}_2(\boldsymbol{k})$ given by

$$\boldsymbol{\mathcal{C}}_2(\boldsymbol{k})\boldsymbol{a} \stackrel{\text{def}}{=} \boldsymbol{k} *_2 \boldsymbol{a}. \tag{6.14}$$

Here, arguments $\boldsymbol{k}$ and $\boldsymbol{a}$ are of the same type as in full convolution.

The difference between valid and full convolution is exemplified in Figure 6.2.

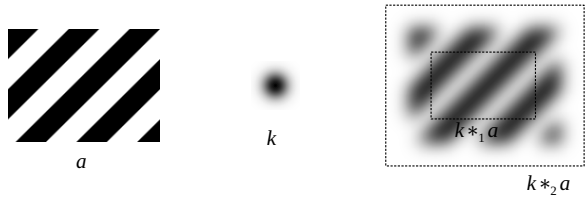


Figure 6.2. Convolving test image $\boldsymbol{a}$ with a Gaussian kernel $\boldsymbol{k}$. The inner rectangle represents valid convolution whereas the outer marks full convolution. Here, black and white correspond to values of 1 and 0, respectively. Kernel $\boldsymbol{k}$ is displayed after min-max normalization.

6.3.2 Adjoint Expressions

As we have seen in the comprehensive benchmark in Subsection 5.2.1, and particularly in Table 5.1, the oracle functions of inverse problems with linear forward models employ the adjoint of the model operator. Our aim is to express our forward model in terms of the aforementioned convolution operators. Therefore, we need to derive the adjoints of these convolution operators.

Adjoint of circular convolution
To be able to express the adjoint of the circular convolution operator, we define the circular shift operator $\boldsymbol{\mathcal{S}}(p, q)$ as

$$\boldsymbol{\mathcal{S}}(p, q)\bar{\boldsymbol{a}} \stackrel{\text{def}}{=} \boldsymbol{e}(p, q) \circledast \bar{\boldsymbol{a}}, \tag{6.15}$$

where $e(p, q)$ is the standard basis vector image

$$\boldsymbol{e}(p, q)_{i,j} = \begin{cases} 1, & i = p,\ j = q, \\ 0, & \text{otherwise.} \end{cases} \tag{6.16}$$

It follows that $\boldsymbol{\mathcal{S}}(1, 1)$ is the identity operator.

The derivation of the adjoint relies on the following basic property of the shift operators.

Lemma 11. *Shift operators cumulate and can be taken out of convolutions as*

$$\boldsymbol{\mathcal{C}}(\boldsymbol{\mathcal{S}}(p_1, q_1)\bar{\boldsymbol{k}})\boldsymbol{\mathcal{S}}(p_2, q_2) = \boldsymbol{\mathcal{S}}(p_1 \oplus_m p_2, q_1 \oplus_n q_2)\boldsymbol{\mathcal{C}}(\bar{\boldsymbol{k}}).$$

Proof. The result follows trivially from the commutativity and associativity properties of circular convolution

$$\begin{aligned} \boldsymbol{\mathcal{C}}(\boldsymbol{\mathcal{S}}(p_1, q_1)\bar{\boldsymbol{k}})\boldsymbol{\mathcal{S}}(p_2, q_2)\bar{\boldsymbol{a}} &= (\boldsymbol{e}(p_1, q_1) \circledast \bar{\boldsymbol{k}}) \circledast (\boldsymbol{e}(p_2, q_2) \circledast \bar{\boldsymbol{a}}) \\ &= (\boldsymbol{e}(p_1, q_1) \circledast \boldsymbol{e}(p_2, q_2)) \circledast \bar{\boldsymbol{k}} \circledast \bar{\boldsymbol{a}} \\ &= \boldsymbol{e}(p_1 \oplus_m p_2, q_1 \oplus_n q_2) \circledast (\bar{\boldsymbol{k}} \circledast \bar{\boldsymbol{a}}) \\ &= \boldsymbol{\mathcal{S}}(p_1 \oplus_m p_2, q_1 \oplus_n q_2)\boldsymbol{\mathcal{C}}(\bar{\boldsymbol{k}})\bar{\boldsymbol{a}}. \end{aligned} \tag{6.17}$$

□

Using notation (6.15) and Lemma 11 we can proceed to the main result.

Theorem 8. *The adjoint of circular convolution is circular cross-correlation circularly shifted forward by one position*

$$(\boldsymbol{\mathcal{C}}(\bar{\boldsymbol{k}}))^T = \boldsymbol{\mathcal{S}}(2, 2)\boldsymbol{\mathcal{C}}(\boldsymbol{\mathcal{R}}(\bar{\boldsymbol{k}})).$$

Proof. From the convolution theorem in (6.6) we have that

$$\bar{\boldsymbol{k}} \circledast \bar{\boldsymbol{a}} = \boldsymbol{F}^H \operatorname{diag}(\boldsymbol{F}\bar{\boldsymbol{k}})\boldsymbol{F}\bar{\boldsymbol{a}}, \tag{6.18}$$

where $\mathrm{diag}(\boldsymbol{x})$ produces a diagonal matrix (linear operator) with the entries of $\boldsymbol{x}$. Therefore, the circular convolution operator is diagonalized in the Fourier domain as

$$\boldsymbol{\mathcal{C}}(\bar{\boldsymbol{k}}) = \boldsymbol{F}^H diag(\boldsymbol{F}\bar{\boldsymbol{k}})\boldsymbol{F}. \tag{6.19}$$

Taking the adjoint in (6.19), we obtain that

$$\begin{aligned}(\boldsymbol{\mathcal{C}}(\bar{\boldsymbol{k}}))^T = (\boldsymbol{\mathcal{C}}(\bar{\boldsymbol{k}}))^H &= \boldsymbol{F}^H diag((\boldsymbol{F}\bar{\boldsymbol{k}})^*)\boldsymbol{F} \\ &= \boldsymbol{F}^H diag(\boldsymbol{F}^*\bar{\boldsymbol{k}})\boldsymbol{F}. \end{aligned} \tag{6.20}$$

The DFT matrix has conjugate symmetry [62], namely

$$\boldsymbol{F}^*\bar{\boldsymbol{k}} = \boldsymbol{F}\boldsymbol{\mathcal{S}}(2,2)\boldsymbol{\mathcal{R}}(\bar{\boldsymbol{k}}). \tag{6.21}$$

Substituting (6.21) in (6.20) we have

$$(\boldsymbol{\mathcal{C}}(\bar{\boldsymbol{k}}))^T = \boldsymbol{F}^H diag(\boldsymbol{F}\boldsymbol{\mathcal{S}}(2,2)\boldsymbol{\mathcal{R}}(\bar{\boldsymbol{k}}))\boldsymbol{F}, \tag{6.22}$$

which can be expressed using the convolution theorem in (6.18) as

$$(\boldsymbol{\mathcal{C}}(\bar{\boldsymbol{k}}))^T = \boldsymbol{\mathcal{C}}(\boldsymbol{\mathcal{S}}(2,2)\boldsymbol{\mathcal{R}}(\bar{\boldsymbol{k}})). \tag{6.23}$$

The application of Lemma 11 with $p_1 = q_1 = 2$, $p_2 = q_2 = 1$, in (6.23) completes the proof. □

Adjoint of valid and full convolution
Circular convolution is a fundamental mathematical construct and Theorem 8 can be used to derive the adjoint of valid and full convolution, as follows.

Theorem 9. *The adjoint of valid convolution is full correlation (full convolution with the rotated kernel), namely*

$$(\boldsymbol{\mathcal{C}}_2(\boldsymbol{k}))^T = \boldsymbol{\mathcal{C}}_1(\boldsymbol{\mathcal{R}}(\boldsymbol{k})).$$

Proof. To simplify notation, we redefine within the scope of this proof the window and zero padding operators, respectively, as

$$\boldsymbol{\mathcal{W}}_{L,H} \overset{\text{def}}{=} \boldsymbol{\mathcal{W}}(m_L, m_H, n_L, n_H, m_N, n_N), \tag{6.24}$$

$$\boldsymbol{\mathcal{Z}}_{L,H} \overset{\text{def}}{=} \boldsymbol{\mathcal{Z}}(m_L, m_H, n_L, n_H, m_N, n_N). \tag{6.25}$$

where indices $L, H \in \{1, k, a, M, N\}$ stand for quantities m_1, m_k, m_a, etc., with $m_1 = n_1 = 1$ and

$$m_M = m_a - m_k + 1, \qquad n_M = n_a - n_k + 1, \tag{6.26}$$

$$m_N = m_a + m_k - 1, \qquad n_N = n_a + n_k - 1. \tag{6.27}$$

Using the above shorthand notation, we can express the valid and full convolution linear operators, respectively, as

$$\boldsymbol{\mathcal{C}}_2(\boldsymbol{k}) = \boldsymbol{\mathcal{W}}_{k,a}\boldsymbol{\mathcal{C}}(\boldsymbol{\mathcal{Z}}_{1,k}\boldsymbol{k})\boldsymbol{\mathcal{Z}}_{1,a}, \tag{6.28}$$

$$\boldsymbol{\mathcal{C}}_1(\boldsymbol{k}) = \boldsymbol{\mathcal{W}}_{1,a}\boldsymbol{\mathcal{C}}(\boldsymbol{\mathcal{Z}}_{1,k}\boldsymbol{k})\boldsymbol{\mathcal{Z}}_{1,M}. \tag{6.29}$$

Note that if $\boldsymbol{\mathcal{C}}_2(\boldsymbol{k})$ takes as arguments images of size $m_a \times n_a$, its adjoint expression involves an operator $\boldsymbol{\mathcal{C}}_1(\boldsymbol{k})$ with arguments of size $m_M \times n_M$. We apply the adjoint in (6.28) and obtain

$$(\boldsymbol{\mathcal{C}}_2(\boldsymbol{k}))^T = (\boldsymbol{\mathcal{Z}}_{1,a})^T(\boldsymbol{\mathcal{C}}(\boldsymbol{\mathcal{Z}}_{1,k}\boldsymbol{k}))^T(\boldsymbol{\mathcal{W}}_{k,a})^T. \tag{6.30}$$

Like their full-width counterparts, the window operator and corresponding zero padding operators introduced in this section are also mutually adjoint, namely

$$(\boldsymbol{\mathcal{W}}_{L,H})^T = \boldsymbol{\mathcal{Z}}_{L,H}. \tag{6.31}$$

Applying (6.31) in (6.30) yields

$$(\boldsymbol{\mathcal{C}}_2(\boldsymbol{k}))^T = \boldsymbol{\mathcal{W}}_{1,a}(\boldsymbol{\mathcal{C}}(\boldsymbol{\mathcal{Z}}_{1,k}\boldsymbol{k}))^T\boldsymbol{\mathcal{Z}}_{k,a}. \tag{6.32}$$

Theorem 8 gives

$$\begin{aligned}(\boldsymbol{\mathcal{C}}(\boldsymbol{\mathcal{Z}}_{1,k}\boldsymbol{k}))^T &= \boldsymbol{\mathcal{S}}(2,2)\boldsymbol{\mathcal{C}}(\boldsymbol{\mathcal{R}}(\boldsymbol{\mathcal{Z}}_{1,k}\boldsymbol{k})) \\ &= \boldsymbol{\mathcal{S}}(2,2)\boldsymbol{\mathcal{C}}(\boldsymbol{\mathcal{Z}}_{a,N}\boldsymbol{\mathcal{R}}(\boldsymbol{k})) \\ &= \boldsymbol{\mathcal{S}}(2,2)\boldsymbol{\mathcal{C}}(\boldsymbol{\mathcal{S}}(m_a,n_a)\boldsymbol{\mathcal{Z}}_{a,N}\boldsymbol{\mathcal{R}}(\boldsymbol{k})) \\ &= \boldsymbol{\mathcal{S}}(m_a+1,n_a+1)\boldsymbol{\mathcal{C}}(\boldsymbol{\mathcal{Z}}_{1,k}\boldsymbol{\mathcal{R}}(\boldsymbol{k})).\end{aligned} \tag{6.33}$$

Using (6.33) in (6.32) and applying Lemma 11 we obtain that

$$\begin{aligned}(\boldsymbol{\mathcal{C}}_2(\boldsymbol{k}))^T &= \boldsymbol{\mathcal{W}}_{1,a}\boldsymbol{\mathcal{S}}(m_a+1,n_a+1)\boldsymbol{\mathcal{C}}(\boldsymbol{\mathcal{Z}}_{1,k}\boldsymbol{\mathcal{R}}(\boldsymbol{k}))\boldsymbol{\mathcal{S}}(m_k,n_k)\boldsymbol{\mathcal{Z}}_{1,M} \\ &= \boldsymbol{\mathcal{W}}_{1,a}\boldsymbol{\mathcal{S}}(1,1)\boldsymbol{\mathcal{C}}(\boldsymbol{\mathcal{Z}}_{1,k}\boldsymbol{\mathcal{R}}(\boldsymbol{k}))\boldsymbol{\mathcal{Z}}_{1,M} \\ &= \boldsymbol{\mathcal{W}}_{1,a}\boldsymbol{\mathcal{C}}(\boldsymbol{\mathcal{Z}}_{1,k}\boldsymbol{\mathcal{R}}(\boldsymbol{k}))\boldsymbol{\mathcal{Z}}_{1,M}.\end{aligned} \tag{6.34}$$

The desired result follows by observing that the right-hand sides of (6.34) and (6.29) are identical. □

Corollary 1. *The adjoint of full convolution is valid correlation (valid convolution with the rotated kernel), namely*

$$(\boldsymbol{\mathcal{C}}_1(\boldsymbol{k}))^T = \boldsymbol{\mathcal{C}}_2(\boldsymbol{\mathcal{R}}(\boldsymbol{k})).$$

Proof. Just like transposition, the rotation operator is unitary and self-adjoint, namely

$$\boldsymbol{\mathcal{R}}(\boldsymbol{\mathcal{R}}(\boldsymbol{k})) = \boldsymbol{k}, \tag{6.35}$$

for all kernels $\boldsymbol{k}$. The rotation operator is also bijective. Therefore, rotating the kernel in Theorem 9 does not reduce its generality. Taking the adjoint in Theorem 9 applied to rotated kernel $\boldsymbol{\mathcal{R}}(\boldsymbol{k})$, we have that

$$\boldsymbol{\mathcal{C}}_2(\boldsymbol{\mathcal{R}}(\boldsymbol{k})) = (\boldsymbol{\mathcal{C}}_1(\boldsymbol{\mathcal{R}}(\boldsymbol{\mathcal{R}}(\boldsymbol{k}))))^T. \tag{6.36}$$

Using (6.35) in (6.36) completes the proof. □

6.4 Ultrasound Image Formation Models

In this section we provide two increasingly accurate models for the formation of ultrasound radio-frequency images.

6.4.1 Prototype Mixture Model

We propose the following image formation model

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{P}\boldsymbol{x} + \boldsymbol{n}, \tag{6.37}$$

where $\boldsymbol{y}$ denotes the observed radio-frequency (RF) image, $\boldsymbol{x}$ is the tissue reflectivity function (TRF) to be recovered and $\boldsymbol{n}$ represents independent identically distributed (i.i.d.) additive white Gaussian noise (AWGN). All images are of the same size, $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{n} \in \mathcal{D}_f \stackrel{\text{def}}{=} \mathbb{R}^{m_t \times n_t}$, where m_t denotes the height (number of pixels along the axial dimension) and n_t gives the width (lateral pixel count) of the TRF.

Padding
Operator $\boldsymbol{P} : \mathbb{R}^{m_t \times n_t} \to \mathbb{R}^{m_p \times n_p}$ pads the TRF with a boundary of width n_r and height m_r, yielding an image of size $m_p = m_t + 2m_r$ times $n_p = n_t + 2n_r$. Padding in our ultrasound imaging model allows us to reconstruct a TRF of the same size as the observed RF image. To this end, we must simulate the effect the surrounding tissues have on the imaged tissues. With padding, we estimate the TRF from the surrounding tissues using information from the imaged TRF. This estimation only affects the border of the reconstructed TRF. If this border information is not required, the reconstructed TRF can simply be cropped accordingly. The addition of padding to our model brings the advantage of accommodating both options.

For computational reasons, $\boldsymbol{P}$ is assumed linear and separable along the dimensions of the image. Separability translates to $\boldsymbol{P} = \boldsymbol{P}_m\boldsymbol{P}_n$. Here, $\boldsymbol{P}_m$ pads every column of the image independently by applying the 1D padding (linear) operator $\boldsymbol{\mathcal{P}}(m_t, m_r)$. Consequently, when $n_t = 1$ and $n_r = 0$, operators $\boldsymbol{P}$ and $\boldsymbol{\mathcal{P}}(m_t, m_r)$ are equivalent. The row component $\boldsymbol{P}_n$ treats every row as a column vector, applies $\boldsymbol{\mathcal{P}}(n_t, n_r)$ to it, and turns the result back into a row.

Padding, either in 1D or 2D, can be performed without explicitly deriving an operator matrix. However, the matrix form facilitates the formulation of the corresponding adjoint operator. Common matrix forms of operator $\boldsymbol{\mathcal{P}}(m_t, n_t)$ are shown in Figure 6.3 for $m_t = 10$ and $m_r = 3$. These examples demonstrate that the matrix form of $\boldsymbol{\mathcal{P}}(m_t, m_r)$ can be easily generated programatically and, due to its sparsity, can be stored in memory even for very large values of m_t and m_r. These properties extend to the matrix form of the 2D padding operator $\boldsymbol{P}$ by virtue of the following result.

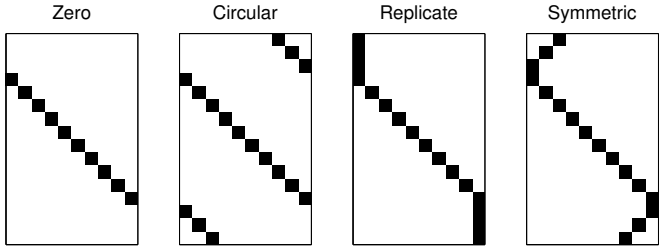


Figure 6.3. Common matrix forms of 1D padding operators $\boldsymbol{\mathcal{P}}(10, 3)$. Black denotes a value of 1 and white denotes 0.

Theorem 10. *Padding operator $\boldsymbol{P}$ can be obtained programatically in the form of a sparse matrix as*

$$\boldsymbol{P} = \boldsymbol{\mathcal{P}}(n_t, n_r) \otimes \boldsymbol{\mathcal{P}}(m_t, m_r).$$

Proof. Since column-major vectorization stacks image columns one on top of the other, $\boldsymbol{P}_m$ will have a block diagonal structure, with each block given by $\boldsymbol{\mathcal{P}}(m_t, m_r)$, namely

$$\begin{aligned} \boldsymbol{P}_m &= \text{diag}\{\boldsymbol{\mathcal{P}}(m_t, m_r), \boldsymbol{\mathcal{P}}(m_t, m_r), ..., \boldsymbol{\mathcal{P}}(m_t, m_r)\} \\ &= \boldsymbol{I}_{n_t} \otimes \boldsymbol{\mathcal{P}}(m_t, m_r). \end{aligned} \tag{6.38}$$

Matrix $\boldsymbol{P}_n$ lacks this block structure. It instead has the elements of $\boldsymbol{\mathcal{P}}(n_t, n_r)$ strided horizontally by m_t and replicated along the diagonal. This can be expressed succinctly as

$$\begin{aligned} \boldsymbol{P}_m &= \begin{bmatrix} \boldsymbol{\mathcal{P}}(n_t, n_r)_{1,1}\boldsymbol{I}_{m_t} & \cdots & \boldsymbol{\mathcal{P}}(n_t, n_r)_{1,n_t}\boldsymbol{I}_{m_t} \\ \vdots & \ddots & \vdots \\ \boldsymbol{\mathcal{P}}(n_t, n_r)_{m_p,1}\boldsymbol{I}_{m_t} & \cdots & \boldsymbol{\mathcal{P}}(n_t, n_r)_{m_p,n_t}\boldsymbol{I}_{m_t} \end{bmatrix} \\ &= \boldsymbol{\mathcal{P}}(n_t, n_r) \otimes \boldsymbol{I}_{m_t}. \end{aligned} \tag{6.39}$$

By combining the results along the dimensions in (6.38) and (6.39), we can compute $\boldsymbol{\mathcal{P}}$ in closed form as

$$\begin{aligned} \boldsymbol{P} = \boldsymbol{P}_m\boldsymbol{P}_n &= (\boldsymbol{I}_{n_t} \otimes \boldsymbol{\mathcal{P}}(m_t, m_r))(\boldsymbol{\mathcal{P}}(n_t, n_r) \otimes \boldsymbol{I}_{m_t}) \\ &= \boldsymbol{\mathcal{P}}(n_t, n_r) \otimes \boldsymbol{\mathcal{P}}(m_t, m_r). \end{aligned} \tag{6.40}$$

□

Prototype mixture convolution

Linear operator $\boldsymbol{H} : \mathbb{R}^{m_p \times n_p} \to \mathbb{R}^{m_t \times n_t}$ performs the prototype mixture convolution. We assume that a small number n_k of prototype kernels is known, each prototype PSF $\boldsymbol{k}_q$, $q \in \{1, ..., n_k\}$, having a center at row c_q. The prototype PSFs are sorted by c_q and thus the values of c_q can be used

to generate a partition of the set of rows $i \in \{1, ..., m_t\}$. The kernels of each row are computed using linear interpolation. Specifically, kernels above c_1 are equal to $\boldsymbol{k}_1$, those below c_{n_k} equal $\boldsymbol{k}_{n_k}$. The kernels of all other rows are obtained as a convex combination (alpha-blending) of the prototype PSF above and the one below that row, the proportion given by the relative distance to the two centers.

Let the full-width window and zero padding operators be defined as

$$\boldsymbol{\mathcal{W}}_s(i_1, i_2) \stackrel{\text{def}}{=} \boldsymbol{\mathcal{W}}(i_1, i_2, 1, n_s, m_s, n_s), \tag{6.41}$$

$$\boldsymbol{\mathcal{Z}}_s(i_1, i_2) \stackrel{\text{def}}{=} \boldsymbol{\mathcal{Z}}(i_1, i_2, 1, n_s, m_s, n_s). \tag{6.42}$$

where $j \in \{1, ..., n_s\}$ and index $s \in \{t, p\}$ stands for image size quantities m_t, m_p, n_t, and n_p.

Using the full width operators, linear operator $\boldsymbol{H}$ can be expressed analytically in a matrix-free form as

$$\begin{aligned} \boldsymbol{H} = {} & \boldsymbol{\mathcal{Z}}_y(1, c_1)\mathcal{C}_1(\boldsymbol{k}_1)\boldsymbol{\mathcal{W}}_p(1, c_1 + 2m_r) \\ & + \sum_{q=1}^{n_k-1} \sum_{i=c_q+1}^{c_{q+1}} \boldsymbol{\mathcal{Z}}_y(i, i)\mathcal{C}_1(\boldsymbol{k}(i, q))\boldsymbol{\mathcal{W}}_p(i, i + 2m_r) \\ & + \boldsymbol{\mathcal{Z}}_y(c_{n_k} + 1, m_y)\mathcal{C}_1(\boldsymbol{k}_{n_k})\boldsymbol{\mathcal{W}}_p(c_{n_k} + 1, m_p), \end{aligned} \tag{6.43}$$

where the row kernels $\boldsymbol{k}(i, q)$ and the row blending factors $\theta(i, q)$ are, respectively, given by

$$\boldsymbol{k}(i, q) \stackrel{\text{def}}{=} (1 - \theta(i, q))\boldsymbol{k}_q + \theta(i, q)\boldsymbol{k}_{q+1}, \tag{6.44}$$

$$\theta(i, q) \stackrel{\text{def}}{=} \frac{i - c_q}{c_{q+1} - c_q}. \tag{6.45}$$

Adjoint

The objective function of the inverse problem contains a data fidelity term $\phi(\boldsymbol{HPx} - \boldsymbol{y})$. Unlike the forward model which, by utilizing operators $\boldsymbol{H}$ and $\boldsymbol{P}$, can be computed exactly with great efficiency, black-box first-order methods such as ACGM employ at every iteration the gradient of the data fidelity term, given by

$$\nabla(\phi(\boldsymbol{HPx} - \boldsymbol{y})) = \boldsymbol{P}^T\boldsymbol{H}^T(\nabla\phi)(\boldsymbol{HPx} - \boldsymbol{y}). \tag{6.46}$$

Note that, under our AWGN assumption, ϕ is the square of the ℓ_2-norm but the results in this work may be applied to other additive noise models.

The gradient expression in (6.46) depends on $\boldsymbol{H}$ and $\boldsymbol{P}$ as well as on their adjoints. In the following, we derive computationally efficient expressions for adjoint operators $\boldsymbol{H}^T$ and $\boldsymbol{P}^T$.

To obtain the adjoint of prototype mixture convolution, we take the

adjoint in (6.43) and get

$$\begin{aligned}\boldsymbol{H}^T &= (\boldsymbol{\mathcal{W}}_p(1, c_1 + 2m_r))^T(\mathcal{C}_2(\boldsymbol{k}_1))^T(\boldsymbol{\mathcal{Z}}_y(1, c_1))^T \\ &+ \sum_{q=1}^{n_k-1}\sum_{i=c_q+1}^{c_{q+1}} (\boldsymbol{\mathcal{W}}_p(i, i + 2m_r))^T(\mathcal{C}_2(\boldsymbol{k}(i, q)))^T(\boldsymbol{\mathcal{Z}}_y(i, i))^T \\ &+(\boldsymbol{\mathcal{W}}_p(c_{n_k} + 1, m_p))^T(\mathcal{C}_2(\boldsymbol{k}_{n_k}))^T(\boldsymbol{\mathcal{Z}}_y(c_{n_k} + 1, m_y))^T.\end{aligned} \tag{6.47}$$

Using (6.5) and Theorem 9 in (6.47), we obtain the matrix-free expression

$$\begin{aligned}\boldsymbol{H}^T &= \boldsymbol{\mathcal{Z}}_p(1, c_1 + 2m_r)\mathcal{C}_1(\boldsymbol{\mathcal{R}}(\boldsymbol{k}_1))\boldsymbol{\mathcal{W}}_y(1, c_1) \\ &+ \sum_{q=1}^{n_k-1}\sum_{i=c_q+1}^{c_{q+1}} \boldsymbol{\mathcal{Z}}_p(i, i + 2m_r)\mathcal{C}_1(\boldsymbol{\mathcal{R}}(\boldsymbol{k}(i, q)))\boldsymbol{\mathcal{W}}_y(i, i) \\ &+\boldsymbol{\mathcal{Z}}_p(c_{n_k} + 1, m_p)\mathcal{C}_1(\boldsymbol{\mathcal{R}}(\boldsymbol{k}_{n_k}))\boldsymbol{\mathcal{W}}_y(c_{n_k} + 1, m_y).\end{aligned} \tag{6.48}$$

The adjoint of the padding operator, $\boldsymbol{P}^T$, can be obtained either directly through sparse matrix transposition or by applying transposition in Theorem 10 as

$$\boldsymbol{P}^T = (\boldsymbol{P})^T = (\boldsymbol{\mathcal{P}}(n_t, n_r))^T \otimes (\boldsymbol{\mathcal{P}}(m_t, m_r))^T. \tag{6.49}$$

Finally, note that whereas the column-major order assumption can be made without loss of generality for operator $\boldsymbol{H}$, it is not the case for the padding operator $\boldsymbol{P}$. In particular, row-major order reverses the terms in the Kronecker product.

Inverse problem

The proposed image formation model in (6.37) can be used to construct a deconvolution problem which seeks to minimize the additive white Gaussian noise subject to regularization. To regularize the solution of this least-squares problem, we use herein the classical elastic net constraint [65]. Elastic net ensures a compromise between the ℓ_1-norm promoting sparse solutions and the ℓ_2-norm imposing smooth results. Its interest in ultrasound imaging has been already shown through different applications, e.g., compressed sensing [66], beamforming [67], or clutter filtering [68]. The inverse problem is given by

$$\min_{\boldsymbol{x}\in\mathcal{D}_f} \frac{1}{2}\|\boldsymbol{H}\boldsymbol{P}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda_1\|\boldsymbol{x}\|_1 + \frac{\lambda_2}{2}\|\boldsymbol{x}\|_2^2. \tag{6.50}$$

6.4.2 Axially Variant Kernel Model

The prototype mixture model can be generalized to yield an axially variant kernel model, whereby a prototype kernel is assigned to every row of the image. The features of the prototype mixture model introduced in (6.37) carry over to this new model, with the exception of the way the axially variant convolution operator $\boldsymbol{H}$ is defined. In particular, padding and its adjoint as well as the inverse problem in (6.50) apply here as well.

Axially varying convolution
We define axially variant convolution as the linear operation whereby each *row* $i_h \in \{1, ..., m_t\}$ of the output image is obtained by the valid convolution between the kernel pertaining to that row $\boldsymbol{k}(i_h) \in \mathbb{R}^{m_k \times n_k}$, where $m_k = 2m_r + 1$ and $n_k = 2n_r + 1$, and the corresponding patch in the input (padded) TRF. The auxiliary operators defined in Sections 6.2 and 6.3 enable us to write $\boldsymbol{H}$ as a sum of linear operators based on the observation that the concatenation of output rows has the same effect as the summation of the rows appropriately padded with zeros. Analytically, this translates to the following matrix-free expression:

$$\boldsymbol{H} = \sum_{i_h=1}^{m_t} \boldsymbol{\mathcal{Z}}_t(i_h, i_h)\boldsymbol{\mathcal{C}}_2(\boldsymbol{k}(i_h))\boldsymbol{\mathcal{W}}_p(i_h, i_h + 2m_r). \tag{6.51}$$

In matrix form, operator $\boldsymbol{H}$ would need to store $m_p n_p m_t n_t$ coefficients and its invocation would entail an equal number of multiplications. Its complexity would thus be greater than the square of the number of pixels in the image, limiting its applicability to medium sized images. Using the matrix-free expression in (6.51), operator $\boldsymbol{H}$ performs $m_k n_k m_t n_t$ multiplications and has negligible memory requirements. Therefore, in ultrasound imaging, the matrix-free representation is not only vastly superior to its matrix counterpart (because the kernel is much smaller than the image), but has the same computational complexity as spatially invariant convolution (excluding the unrealistic circulant boundary case).

Adjoint
By taking the adjoint in (6.51), we get

$$\boldsymbol{H}^T = \sum_{i_h=1}^{m_t} (\boldsymbol{\mathcal{W}}_p(i_h, i_h + 2m_r))^T (\boldsymbol{\mathcal{C}}_2(\boldsymbol{k}(i_h)))^T (\boldsymbol{\mathcal{Z}}_t(i_h, i_h))^T. \tag{6.52}$$

Theorem 9 and (6.5) yield a matrix-free expression for $\boldsymbol{H}^T$ in the form of

$$\boldsymbol{H}^T = \sum_{i_h=1}^{m_t} \boldsymbol{\mathcal{Z}}_p(i_h, i_h + 2m_r)\boldsymbol{\mathcal{C}}_1(\boldsymbol{\mathcal{R}}(\boldsymbol{k}(i_h)))\boldsymbol{\mathcal{W}}_t(i_h, i_h). \tag{6.53}$$

Therefore, operators $\boldsymbol{H}$ and $\boldsymbol{H}^T$ have equal computational complexity. Moreover, they exhibit two levels of parallelism. The convolution operators themselves are fully parallel and the computations pertaining to each row i_h can be performed concurrently. Thus, in matrix-free from, both operators benefit from parallelization in the same way as their matrix counterparts.

6.5 Optimizing ACGM for Linear Inverse Problems

To efficiently solve the optimization problem in (6.50) for any non-negative value of λ_2, we optimize ACGM for linear inverse problems.

The objective F in problem (6.50) can be split into a quadratic function f and an elastic net regularizer Ψ as follows:

$$f(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2, \tag{6.54}$$

$$\Psi(\boldsymbol{x}) = \lambda_1\|\boldsymbol{x}\|_1 + \frac{\lambda_2}{2}\|\boldsymbol{x}\|_2^2, \tag{6.55}$$

where $\boldsymbol{A} \stackrel{\text{def}}{=} \boldsymbol{H}\boldsymbol{P}$. Function f is quadratic and consequently has Lipschitz continuous gradient. The Lipschitz constant L_f is given by $\sigma_{max}^2(\boldsymbol{A})$, where $\sigma_{max}(\boldsymbol{A})$ is the largest singular value of operator $\boldsymbol{A}$. In practice, $\sigma_{max}(\boldsymbol{A})$ may be intractable to compute. Hence, this problem is well suited for ACGM. However it is known that operator $\boldsymbol{A}$ is ill-conditioned and we can assume that function f has strong convexity parameter $\mu_f = 0$. Elastic net regularizer Ψ is not differentiable due to the l_1 term but has strong convexity parameter $\mu_\Psi = \lambda_2$. Hence, the objective as a whole has a strong convexity parameter of $\mu = \mu_\Psi = \lambda_2$.

The gradient-type oracle functions $\nabla f(\boldsymbol{x})$ and $\text{prox}_{\tau\Psi}(\boldsymbol{x})$ are, respectively, written in closed form (see also [18] and Subsection 5.2.1) as

$$\nabla f(\boldsymbol{x}) = \boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}), \tag{6.56}$$

$$\text{prox}_{\tau\Psi}(\boldsymbol{x}) = \frac{1}{1+\tau\lambda_2}\mathcal{T}_{\tau\lambda_1}(\boldsymbol{x}), \tag{6.57}$$

where $\boldsymbol{A}^T = \boldsymbol{P}^T\boldsymbol{H}^T$ and the shrinkage operator $\mathcal{T}_\tau(\boldsymbol{x})$ is given by

$$\begin{gathered}(\mathcal{T}_\tau(\boldsymbol{x}))_{i,j} \stackrel{\text{def}}{=} (|\boldsymbol{x}_{i,j}| - \tau)_+ \operatorname{sgn}(\boldsymbol{x}_{i,j}), \\ \tau > 0, \quad i \in \{1, ..., m_t\}, \quad j \in \{1, ..., n_t\}.\end{gathered} \tag{6.58}$$

The structure of $f(\boldsymbol{x})$ in (6.54) and $\nabla f(\boldsymbol{x})$ in (6.56) can be used to reduce the computational complexity of ACGM by departing from the oracle model. To estimate the local Lipschitz constant, operator $\boldsymbol{A}$ has to be applied at every iteration k to the new iterate $\boldsymbol{x}_{k+1}$. It is computationally inexpensive to cache these results by maintaining alongside the main iterate sequence the sequence $\tilde{\boldsymbol{x}}_k$, given by

$$\tilde{\boldsymbol{x}}_k = \boldsymbol{A}\boldsymbol{x}_k, \quad k \in \{-1, ..., K\}. \tag{6.59}$$

ACGM in extrapolated form (Algorithm 6) produces an auxiliary point $\boldsymbol{y}_{k+1}$ (not to be confused with the RF image $\boldsymbol{y}$) as the linear extrapolation of $\boldsymbol{x}_k$ and $\boldsymbol{x}_{k+1}$. The new iterate $\boldsymbol{x}_{k+1}$ is computed based on $\nabla f(\boldsymbol{y}_{k+1})$. The computational intensity of the gradient expression in (6.56) can be reduced as

$$\nabla f(\boldsymbol{y}_{k+1}) = \boldsymbol{A}^T(\tilde{\boldsymbol{y}}_{k+1} - \boldsymbol{y}), \tag{6.60}$$

where

$$\tilde{\boldsymbol{y}}_{k+1} \stackrel{\text{def}}{=} \boldsymbol{A}\boldsymbol{y}_{k+1} = \tilde{\boldsymbol{x}}_k + \beta_k(\tilde{\boldsymbol{x}}_k - \tilde{\boldsymbol{x}}_{k-1}), \tag{6.61}$$

and β_k is the extrapolation factor given by Proposition 4.

The line-search stopping criterion of ACGM at every iteration k is given by (3.15). Substituting the gradient expression from (6.56) in (3.15) and rearranging terms yields

$$\|\boldsymbol{A}(\boldsymbol{x}_{k+1} - \boldsymbol{y}_{k+1})\|_2^2 \leq L_{k+1}\|\boldsymbol{x}_{k+1} - \boldsymbol{y}_{k+1}\|_2^2. \tag{6.62}$$

We obtain a computationally efficient expression by reusing the precomputed values $\tilde{\boldsymbol{x}}_{k+1}$ and $\tilde{\boldsymbol{y}}_{k+1}$ as

$$\|\tilde{\boldsymbol{x}}_{k+1} - \tilde{\boldsymbol{y}}_{k+1}\|_2^2 \leq L_{k+1}\|\boldsymbol{x}_{k+1} - \boldsymbol{y}_{k+1}\|_2^2. \tag{6.63}$$

The global Lipschitz constant L_f can alternatively be expressed as

$$L_f = \sup_{\mathcal{D}_f} \frac{\|\boldsymbol{A}\boldsymbol{x}\|_2^2}{\|\boldsymbol{x}\|_2^2}. \tag{6.64}$$

In practice, the estimates are below this value and we set the initial one to

$$L_0 = \frac{\|\boldsymbol{A}\boldsymbol{x}_0\|_2^2}{\|\boldsymbol{x}_0\|_2^2} = \frac{\|\tilde{\boldsymbol{x}}_0\|_2^2}{\|\boldsymbol{x}_0\|_2^2}. \tag{6.65}$$

Incorporating the performance enhancements from (6.59), (6.60), (6.61), (6.63), and (6.65) into Algorithm 6 yields the method listed in Algorithm 11. To improve performance, we set $t_0 = 0$, as suggested by the findings in Subsection 5.2.3. Note that, to improve readability, we omit the estimate notation in Algorithm 11.

6.6 Experimental Results

6.6.1 Prototype Mixture Model

A three-step process was employed to simulate the RF ultrasound image: i) the calculation of the spatially variant prototype PSFs; ii) the generation of the tissue reflectivity function (TRF); and iii) the spatially variant convolution between the PSFs and the TRF, following the model described in (6.37) (Subsection 6.4.2).

The prototype PSFs were obtained in step i) using Field II, a realistic state-of-the-art simulator [69, 70]. The simulation involved a linear 128 element ultrasound probe emitting ultrasound waves at a nominal frequency of 3 MHz. The width of each element was set to equal the wavelength (0.5 mm), while height was fixed at 5 mm. The distance between two consecutive elements was set to 0.1 mm. The transducer was excited using a two-period sinusoidal wave of frequency 3 MHz. The backscattered

Algorithm 11 ACGM for elastic net regularized least-squares

ACGM$(\boldsymbol{x}_0, r_u, r_d, \lambda_1, \lambda_2, K)$

$\tilde{\boldsymbol{x}}_0 := \boldsymbol{A}\boldsymbol{x}_0$
$\boldsymbol{x}^{(-1)} = \boldsymbol{x}_0$
$\tilde{\boldsymbol{x}}^{(-1)} = \tilde{\boldsymbol{x}}_0$
$L_0 = \frac{\|\tilde{\boldsymbol{x}}_0\|_2^2}{\|\boldsymbol{x}_0\|_2^2}$
$q_0 = \frac{\lambda_2}{L_0+\lambda_2}$
$t_0 = 0$
for $k = 0, ..., K-1$ **do**
  $L_{k+1} := r_d L_k$
  loop
    $q_{k+1} := \frac{\lambda_2}{L_{k+1}+\lambda_2}$
    $t_{k+1} := \frac{1}{2}\left(1 - q_k t_k^2 + \sqrt{\left(1 - q_k t_k^2\right)^2 + 4\frac{L_{k+1}+\lambda_2}{L_k+\lambda_2} t_k^2}\right)$
    $\beta_k := \frac{t_k - 1}{t_{k+1}} \frac{1 - q_{k+1} t_{k+1}}{1 - q_{k+1}}$
    $\boldsymbol{y}_{k+1} := \boldsymbol{x}_k + \beta_k(\boldsymbol{x}_k - \boldsymbol{x}_{k-1})$
    $\tilde{\boldsymbol{y}}_{k+1} := \tilde{\boldsymbol{x}}_k + \beta_k(\tilde{\boldsymbol{x}}_k - \tilde{\boldsymbol{x}}_{k-1})$
    $\tau_{k+1} := \frac{1}{L_{k+1}}$
    $\boldsymbol{x}_{k+1} := \frac{1}{1+\tau_{k+1}\lambda_2} \mathcal{T}_{\tau_{k+1}\lambda_1}\left(\boldsymbol{y}_{k+1} - \tau_{k+1}\boldsymbol{A}^T(\tilde{\boldsymbol{y}}_{k+1} - \boldsymbol{y})\right)$
    $\tilde{\boldsymbol{x}}_{k+1} := \boldsymbol{A}\boldsymbol{x}_{k+1}$
    if $\|\tilde{\boldsymbol{x}}_{k+1} - \tilde{\boldsymbol{y}}_{k+1}\|_2^2 \leq L_{k+1}\|\boldsymbol{x}_{k+1} - \boldsymbol{y}_{k+1}\|_2^2$ **then**
      Break from loop
    else
      $L_{k+1} := r_u L_{k+1}$
    end if
  end loop
end for

RF signals were sampled at a rate of 20 MHz. The prototype PSFs were obtained by placing isolated scatterers in front of the transducer with a distance in depth of 8.5 mm to each other. Ultrasound waves electronically focalized at 47 mm from the probe were emitted and the received echoes were statically focused prior to the delay-and-sum beamforming process. Hann apodization was used both for the emission and the reception.

The resulting $n_k = 10$ prototype PSFs are shown in Figure 6.4. For the purpose of visualizing the areas influenced by individual prototype PSFs, they are displayed after envelope detection and min-max normalization centered at c_q for all $q \in \{1, ..., n_k\}$ in Figure 6.4(a). To highlight the differences between individual prototype PSFs, they are shown separately in Figure 6.4(b). The 5th prototype PSF ($\boldsymbol{k}_5$) located at 43 mm from the probe was used in the spatially invariant deconvolution experiments. It is shown both in native form and after envelope detection in Figure 6.5.

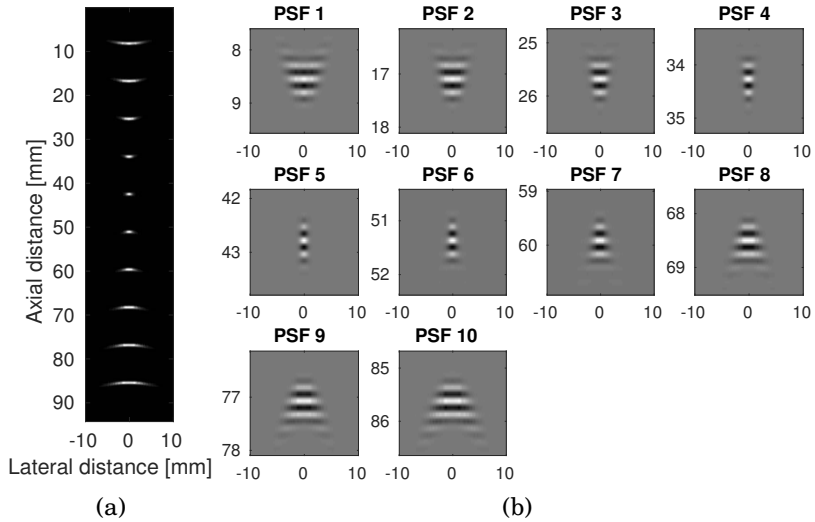


Figure 6.4. Prototype PSFs generated with Field II for $n_k = 10$ depths at regularly spaced intervals of 8.5 mm. (a) Global view, after demodulation and min-max normalization, showing the location within the image of the prototype PSF centers; (b) Individual view, showing the spatial variance of the prototype PSFs.

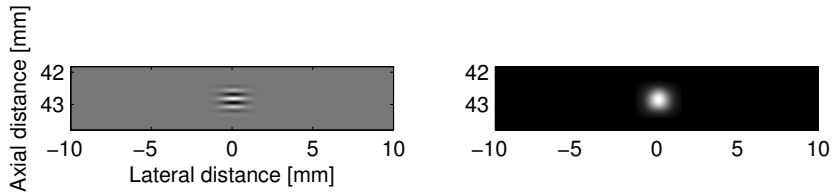


Figure 6.5. The 5th prototype PSF $\boldsymbol{k}_5$ (left) and its demodulated version (right).

The TRF was obtained in step ii) following the classical procedure employed in the simulation of ultrasound images. A collection of scatterers with zero-mean Gaussian random amplitudes has been generated and placed at uniformly random locations. The standard deviation used to generate the amplitude of one scatterer depended on its location and was related, as suggested in the Field II simulator, to a digital image obtained from an optical scan of a human kidney tissue sample. The number of scatterers was sufficiently large (10^5) to ensure fully developed speckle. The scatterer map was finally linearly interpolated onto a rectangular grid yielding the TRF shown in Figure 6.6(a).

In step iii), an ultrasound image was simulated from the TRF using the model in (6.37) to produce a starting point $\boldsymbol{x}_0$ for the deconvolution experiments. First, the TRF was padded with a symmetric boundary. Next,

the padded image was convolved with the spatially varying convolution operator $\boldsymbol{H}$, based on the prototype PSFs shown in Figure 6.4. To simplify the hyperparameter tuning process, we have scaled $\boldsymbol{H}$ to ensure that L_0, as given by (6.65), is equal to 1. Finally, white Gaussian noise was added to the convolved image, such that the signal-to-noise ratio is 40 dB. The simulated ultrasound image is shown in B-mode representation in Figure 6.6(b).

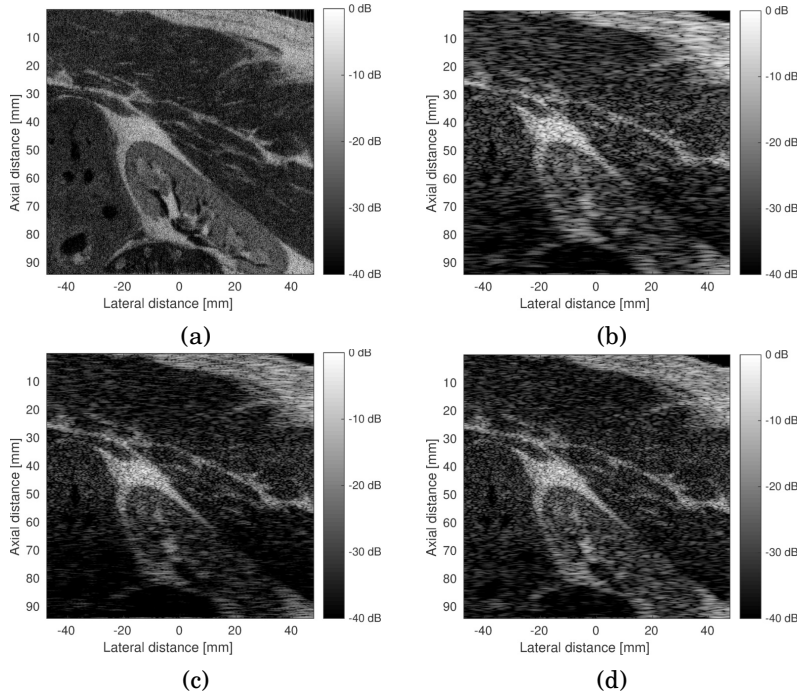


Figure 6.6. (a) Ground truth of the tissue reflectivity function (TRF); (b) Observed B-mode image simulated from the ground truth TRF in (a) using the image acquisition model in (6.37) that employs the spatially variant convolution operation based on the prototype PSFs in Figure 6.4; (c) Spatially invariant deconvolution result obtained using the fixed kernel $\boldsymbol{k}_5$ in Figure 6.5; (d) Spatially variant deconvolution result using our method.

We have conducted two deconvolution experiments. Both used as the starting point the simulated ultrasound image shown in Figure 6.6(b) and the same values of the hyperparameters, $\lambda_1 = 0.005$ and $\lambda_2 = 0.01$, which were found by manual tuning to give the best results.

First, we have compared our method, which is able to integrate the spatial variability of the kernels in the deconvolution process, with ACGM employing a spatially invariant blur operator $\boldsymbol{H}$. Two restored images, obtained after 150 iterations, are displayed in Figures 6.6(c) and 6.6(d). The image in Figure 6.6(c) was estimated considering a spatially invariant PSF

($\boldsymbol{k}_5$ at 43 mm depth) and the one in Figure 6.6(d) was obtained using our method. The quality of the deconvolution can be appreciated by comparing the restored images with the true TRF in Figure 6.6(a). Note that the deconvolution results are also shown in B-mode. While the deconvolved images are similar in the vicinity of the focal point, our method manages to restore image features at the vertical extremities (Figure 6.6(d)). These features are barely discernible both in the blurred image shown (Figure 6.6(b)) as well as in the spatially invariant PSF reconstruction (Figure 6.6(c)). The simulation results support our previous claim that the reconstruction quality of an image patch depends on the similarity between the blurring and the deblurring kernels applied to it, clearly demonstrating the superiority of our model.

6.6.2 Axially Variant Kernel Model

To test the generalized model, we have devised a more extensive set of simulations. Three TRFs (TRF 1, TRF 2, and TRF 3) were generated following the procedure used in step ii) of Subsection 6.6.1. Each TRF is an interpolation of Gaussian distributed random scatterers with standard deviations determined by a pixel intensity map. The map for TRF 1 is the same as the one utilized in Subsection 6.6.1. The maps for TRF 2 and TRF 3 are patches from a single slice (visual identifier 3272) of the female dataset provided by the Visible Human Project [71]. Deconvolution experiments were performed for each of the 3 TRFs, with the padding size, matching the kernel radii, given by $m_r \times n_r$. The TRFs are shown in Figures 6.7(a1), 6.7(a2), and 6.7(a3), respectively, and their parameters are listed in Table 6.1.

Table 6.1. Deconvolution experiment parameters for each TRF

TRF	m_t	n_t	Axial size	Lateral size	m_r	n_r
TRF 1	2480	480	94 mm	95 mm	9	50
TRF 2	2598	480	100 mm	95 mm	7	35
TRF 3	2598	480	100 mm	95 mm	5	25

For every row $i_h \in \{1, ..., m_t\}$, we have defined the kernel $\boldsymbol{k}(i_h)$ in (6.51) as

$$\boldsymbol{k}(i_h)_{i,j} = \rho_{\mu_z,\sigma_z}(i)\rho_{\mu_x,\sigma_x(i_h)}(j)\cos\left(2\pi\frac{f_0}{f_s}(i-\mu_z)\right),$$

where $\rho_{\mu,\sigma}(x)$ is a normalized Gaussian window, given by

$$\rho_{\mu,\sigma}(x) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right),$$

and parameters μ_z and μ_x are the center coordinates of the kernel. Axial

standard deviation (SD) was set to $\sigma_z = \sigma_1$ and lateral SD to

$$\sigma_x(i_h) = \sqrt{((2i_h)/m_t - 1)^2(\sigma_2^2 - \sigma_1^2) + \sigma_1^2},$$

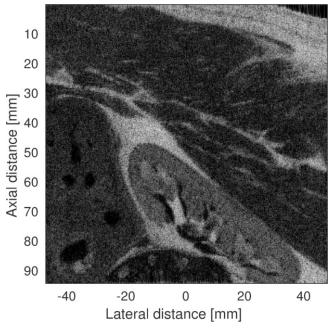
with $\sigma_1 = m_r/3$ and $\sigma_2 = n_r/3$. Here, $f_0 = 3$ MHz and $f_s = 20$ MHz are the ultrasound central and sampling frequencies, respectively. For each TRF deconvolution experiment, the depth-dependent width variation of the kernel simulates the lateral spatial resolution degradation when moving away from the focus point, located in the center of the image (47 mm from the probe for TRF 1, 50 mm for TRF 2 and TRF 3). The envelopes of these kernels at regular intervals across the image are shown in Figures 6.7(b1), 6.7(b2), and 6.7(b3). Whereas the TRFs are of approximately the same size, the intensity of the blur differs in each experiment in order to display both the high and low frequency reconstruction capabilities of our method. We chose symmetric padding, as illustrated in Figure 6.3, because it is more realistic than circular and zero padding and, by using a larger number of pixels from the TRF, more robust to noise than replicate padding. A small amount of noise was added to each TRF such that the signal to noise ratio is 40 dB. The ultrasound images produced from each TRF using our forward model in (6.37) are shown in Figure 6.7(c1), 6.7(c2), and 6.7(c3).

To estimate the TRFs, we have considered the inverse problem in (6.50) with manually tuned parameters $\lambda_1 = 2 \cdot 10^{-3}$ and $\lambda_2 = 10^{-4}$. As in Subsection 6.6.1, we address (6.50) with the optimized variant of ACGM listed in Algorithm 11.

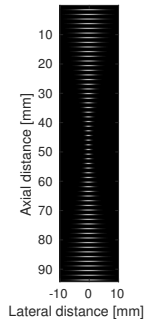
Due to the efficient matrix-free expressions of $\boldsymbol{H}$ in (6.51) and $\boldsymbol{H}^T$ in (6.53) as well as the sparse matrix implementation of $\boldsymbol{P}$ and $\boldsymbol{P}^T$ (easily precomputed using Theorem 10 and (6.49), respectively), deconvolution with our model entails the same computational cost as with a fixed kernel model.

The results of axially-invariant deconvolution (AI) are shown in Figures 6.7(d1), 6.7(d2), and 6.7(d3) and using our axially variant model (AV) in Figures 6.7(e1), 6.7(e2), and 6.7(e3), all after 150 iterations. Our approach achieves almost perfect low frequency reconstruction for each TRF. For higher frequencies, blurring destroys visual information. Therefore, reconstruction quality at this level depends on the size of the kernels. For TRF 3, axial standard deviation values are the smallest and the blurring process has not suppressed all minor details. Here, our method manages to reconstruct even small, high contrast features.

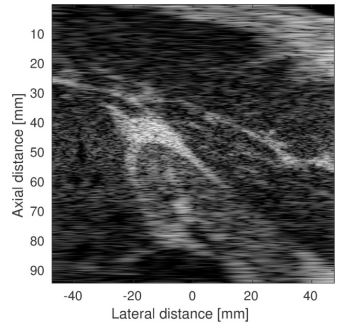
For all TRFs, the gain in reconstruction quality is evident especially in the upper and lower extremities, as can be discerned from Figure 6.7. Interestingly, even though the two models differ only slightly in the center of the image, our model performs better in that region as well.



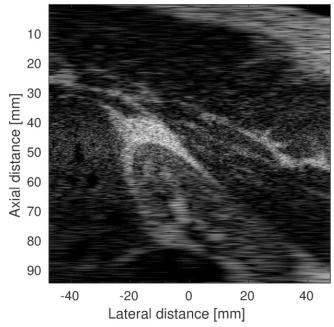
(a1)



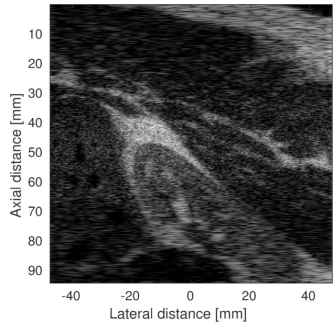
(b1)



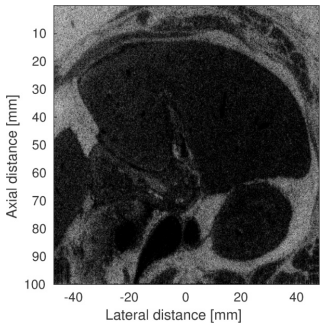
(c1)



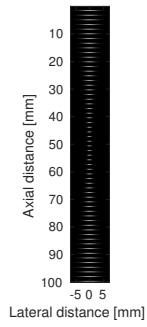
(d1)



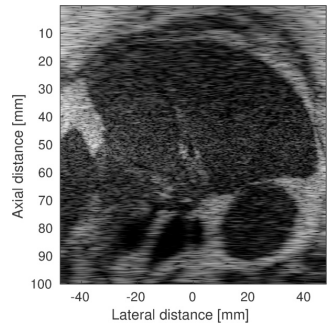
(e1)



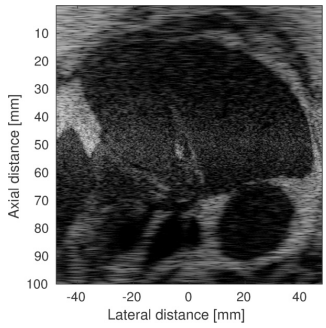
(a2)



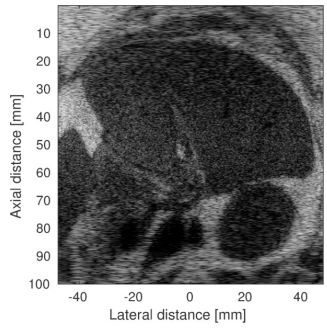
(b2)



(c2)



(d2)



(e2)

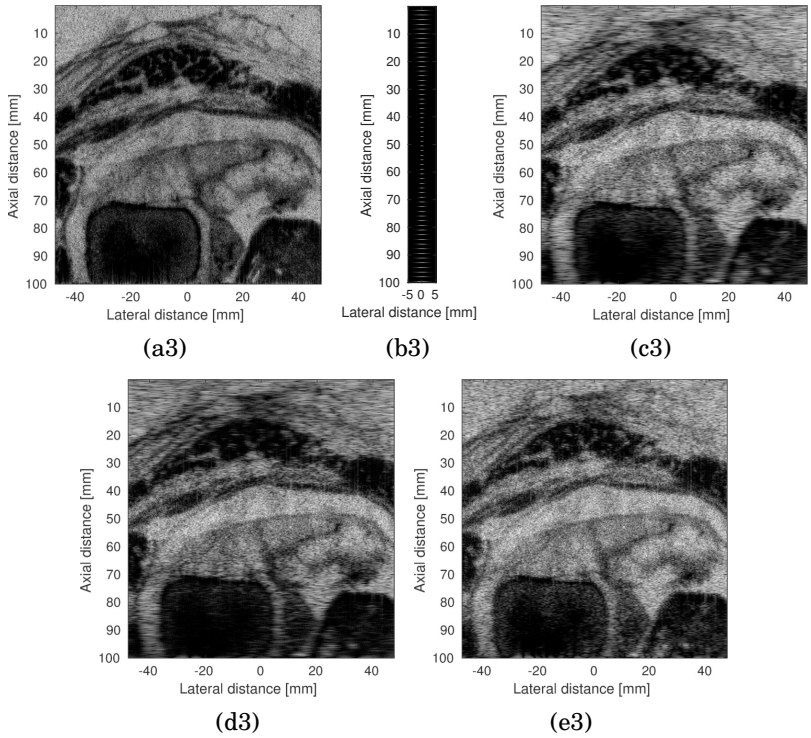


Figure 6.7. (a) Ground truth (in B-mode) of the tissue reflectivity function (TRF); (b) Demodulated kernels $\boldsymbol{k}(i_h)$ for twenty depths at regularly spaced intervals of 2 mm; (c) Observed B-mode image simulated following the proposed axially variant convolution model; (d) Axially-invariant deconvolution result AI (in B-mode) obtained with a fixed kernel equal to $\boldsymbol{k}(m_t/2)$ (the center kernel of the axially variant model); (e) Axially variant deconvolution result AV (in B-mode) using our model. All images are displayed using a dynamic range of 40 dB. Row (a1)-(e1) pertains to TRF 1, row (a2)-(e2) to TRF 2, and row (a3)-(e3) to TRF 3.

7. Conclusions

In this work, we have demonstrated that the estimate sequence can be derived as a functional extension of the theoretical algorithm performance bounds on large scale smooth problems. We have further extended these results to the composite problem class. The estimate sequence optimum is thus a computable substitute for the convergence guarantees on composite problems. Next, we have shown that we can relax the estimate sequence, by reducing the gap between the aforementioned optimum and the convergence guarantee, to obtain the augmented estimate sequence.

The estimate sequence along with its augmented variant incorporate convex global lower bounds on the objective function. When these lower bounds are generalized parabolae, we have shown that the augmented estimate sequence can be expressed using the simple gap sequence. The Lyapunov property of the gap sequence is a sufficient condition, and thus a replacement, for the augmented estimate sequence and, in turn, for the convergence guarantee.

Using the ideas developed by Nesterov, we have formulated a design pattern for first-order accelerated algorithms applicable to composite problems. Composite problems can be defined in terms of local upper bounds and global lower bounds on the objective. The proposed pattern is based on these properties as well as on an accumulated history of all previous global lower bounds combined using a computationally inexpensive weighting strategy.

The design pattern simplifies the derivation of a first-order optimization method with a provable convergence rate to the selection of upper and lower bounds along with the derivation of the corresponding update rules to satisfy the Lyapunov property of the gap sequence. By selecting different types of upper and lower bounds, we were able to derive several variants of the novel Accelerated Composite Gradient Method, each possessing a collection of desirable capabilities. These include optimal convergence guarantees for non-strongly convex objectives, state-of-the-art guarantees for strongly convex objectives, and monotonicity.

Interestingly, the local upper bound property in our pattern and the local

Lipschitz property of the gradient are identical notions. By taking this into account by default, the design process produces a fully adaptive line-search procedure applicable to all the above variants. A further coincidence is that when dealing with arbitrary strong convexity, the auxiliary point and the recent iterates are collinear, a property that is retained to a certain extent even by the monotone variants.

These features are not mutually exclusive and ACGM can posses all of them simultaneously. Thus, ACGM acts as an umbrella method, effectively encompassing several popular first-order schemes: original FGM [16], line-search FGM [24], original FISTA [5], line-search FISTA [38, 39], monotone FISTA [22], as well as strongly convex extensions FISTA-CP [6], and scAPG [21].

Standard WTU makes it possible to compare a wide range of algorithms with vastly varying per-iteration complexity. It is particularly suited for comparing the theoretical performance of FISTA and AMGS which, to the best of our knowledge, has not been attempted before. When measured in standard WTU, ACGM surpasses FISTA with backtracking line-search as well as the algorithms from the AMGS family: original AMGS [17], MOS, and AA [20]. The superior performance of ACGM is demonstrated theoretically and is supported by simulation results. Moreover, simulations show that the standard WTU results also hold for a common form of generalized WTU.

Each of the aforementioned competing methods possesses one characteristic of ACGM, but ACGM is unique among its class of algorithms for having almost all the features of the competing methods. This explains the superiority of ACGM with its class of algorithms.

ACGM can enable new applications, such as the accurate reconstruction of ultrasound images. We have shown that ACGM is able to exploit the inherent structure of the ultrasound image reconstruction inverse problem and is theoretically guaranteed to converge linearly for any type of input data. Simulation results confirm that reconstruction is not only computationally tractable, but has a rate that is competitive with existing approaches relying on far more restrictive assumptions. In this respect, the reliability of ACGM is particularly of value to the stringent demands of the medical ultrasonography industry.

The results presented in this work also open up new avenues for investigation. For instance, we have demonstrated that, under certain conditions, the augmentation of the estimate sequence leads to the gap sequence. New algorithms may be created by studying the effect of augmentation on other types of global lower bounds or even other problem classes.

In the area of ultrasound imaging, ACGM can be used with more complex models. These may include 3D models, non-linear models, or both.

References

[1] P. L. Combettes and J.-C. Pesquet, *Proximal Splitting Methods in Signal Processing*. Springer New York, NY, USA, 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 Processing Magazine*, 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 Processing Magazine*, vol. 31, no. 5, pp. 32–43, Sep. 2014.

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

[5] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, 2009.

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

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

[8] J.-F. Aujol and C. Dossal, "Stability of over-relaxations for the forward-backward algorithm, application to FISTA," *SIAM Journal on Optimization*, pp. 2408–2433, 2015.

[9] R. Gu and A. Dogandžić, "Projected Nesterov's proximal-gradient algorithm for sparse signal recovery," *IEEE Transactions on Signal Processing*, vol. 65, no. 13, pp. 3510–3525, Jul. 2017.

[10] W. W. Hager, D. T. Phan, and H. Zhang, "Gradient-based methods for sparse recovery," *SIAM Journal on Imaging Sciences*, 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 Journal on Scientific Computing*, 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," *Optimization Methods and Software*, 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 Transactions on Signal Processing*, vol. 57, no. 7, pp. 2479–2493, 2009.

[14] A. Nemirovski and D.-B. Yudin, *Problem complexity and method efficiency in optimization*. John Wiley & Sons, New York, NY, USA, 1983.

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

[16] ——, *Introductory Lectures on Convex Optimization. Applied Optimization, vol. 87*. Kluwer Academic Publishers, Boston, MA, USA, 2004.

[17] ——, "Gradient methods for minimizing composite functions," *Mathematical Programming, Series B*, vol. 140, no. 1, pp. 125–161, 2013.

[18] N. Parikh, S. P. Boyd *et al.*, "Proximal algorithms," *Foundations and Trends in Optimization*, vol. 1, no. 3, pp. 127–239, 2014.

[19] P. Tseng, "On accelerated proximal gradient methods for convex-concave optimization," *SIAM Journal on Optimization*, submitted, 2008.

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

[21] Q. Lin and L. Xiao, "An adaptive accelerated proximal gradient method and its homotopy continuation for sparse optimization," in *International Conference on Machine Learning*, 2014, pp. 73–81.

[22] A. Beck and M. Teboulle, "Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems," *IEEE Transactions on Image Processing*, vol. 18, no. 11, pp. 2419–2434, 2009.

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

[24] Y. Nesterov and S. U. Stich, "Efficiency of the accelerated coordinate descent method on structured optimization problems," *SIAM Journal on Optimization*, vol. 27, no. 1, pp. 110–123, 2017.

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

[26] A. Chambolle and C. Dossal, "On the convergence of the iterates of the fast iterative shrinkage/thresholding algorithm," *Journal of Optimization Theory and Applications*, vol. 166, no. 3, pp. 968–982, 2015.

[27] Y. Nesterov, "Quasi-monotone subgradient methods for nonsmooth convex minimization," *Journal of Optimization Theory and Applications*, no. 165, pp. 917–940, 2015.

[28] ——, "Accelerating the cubic regularization of Newton's method on convex problems," *Mathematical Programming*, vol. 112, no. 1, pp. 159–181, 2008.

[29] B. Polyak, *Introduction to optimization*. Translations Series in Mathematics and Engineering, Optimization Software, New York, NY, USA, 1987.

[30] A. Brown and M. C. Bartholomew-Biggs, "Some effective methods for unconstrained optimization based on the solution of systems of ordinary differential equations," *Journal of Optimization Theory and Applications*, vol. 62, no. 2, pp. 211–224, 1989.

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

[32] J. M. Ortega and W. C. Rheinboldt, *Iterative solution of nonlinear equations in several variables*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1970.

[33] K. Lange, *MM optimization algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2016.

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

[35] R. T. Rockafellar, *Convex Analysis*. Princeton University Press, Princeton, NJ, USA, 1970.

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

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

[38] M. I. Florea and S. A. Vorobyov, "A robust FISTA-like algorithm," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, Mar. 2017, New Orleans, USA, pp. 4521–4525.

[39] K. Scheinberg, D. Goldfarb, and X. Bai, "Fast first-order methods for composite convex optimization with backtracking," *Foundations of Computational Mathematics*, vol. 14, no. 3, pp. 389–417, 2014.

[40] R. Tibshirani, "Regression shrinkage and selection via the LASSO," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[41] P. C. Hansen, J. G. Nagy, and D. P. O'Leary, *Deblurring images: matrices, spectra, and filtering*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2006.

[42] D. R. Cox, "The regression analysis of binary sequences," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 20, pp. 215–242, 1958.

[43] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 67, no. 2, pp. 301–320, 2005.

[44] B. O'Donoghue and E. Candès, "Adaptive restart for accelerated gradient schemes," *Foundations of Computational Mathematics*, vol. 15, no. 3, pp. 715–732, 2015.

[45] T. Hastie, R. Tibshirani, and M. Wainwright, *Statistical Learning with Sparsity: The LASSO and Generalizations*. CRC Press, 2015.

[46] W. Su, S. Boyd, and E. J. Candès, "A differential equation for modeling Nesterov's accelerated gradient method: Theory and insights," *Journal of Machine Learning Research (JMLR)*, vol. 17, pp. 1–43, 2016.

[47] J. Ng, R. Prager, N. Kingsbury, G. Treece, and A. Gee, "Wavelet restoration of medical pulse-echo ultrasound images in an EM framework," *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, vol. 54, no. 3, pp. 550–568, 2007.

[48] R. Rangarajan, C. V. Krishnamurthy, and K. Balasubramaniam, "Ultrasonic imaging using a computed point spread function," *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, vol. 55, no. 2, pp. 451–464, Feb. 2008.

[49] H.-C. Shin, R. Prager, J. Ng, H. Gomersall, N. Kingsbury, G. Treece, and A. Gee, "Sensitivity to point-spread function parameters in medical ultrasound image deconvolution," *Ultrasonics*, vol. 49, no. 3, pp. 344 – 357, 2009.

[50] M. Alessandrini, S. Maggio, J. Poree, L. D. Marchi, N. Speciale, E. Franceschini, O. Bernard, and O. Basset, "A restoration framework for ultrasonic tissue characterization," *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, vol. 58, no. 11, pp. 2344–2360, 2011.

[51] C. Dalitz, R. Pohle-Frohlich, and T. Michalk, "Point spread functions and deconvolution of ultrasonic images," *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, vol. 62, no. 3, pp. 531–544, Mar. 2015.

[52] N. Zhao, A. Basarab, D. Kouamé, and J.-Y. Tourneret, "Joint segmentation and deconvolution of ultrasound images using a hierarchical Bayesian model based on generalized Gaussian priors," *IEEE Transactions on Image Processing*, vol. 25, no. 8, pp. 3736–3750, 2016.

[53] O. Michailovich and D. Adam, "A novel approach to the 2-D blind deconvolution problem in medical ultrasound," *IEEE Transactions on Medical Imaging*, vol. 24, pp. 86–104, Jan. 2005.

[54] O. Michailovich and A. Tannenbaum, "Blind deconvolution of medical ultrasound images: A parametric inverse filtering approach," *IEEE Transactions on Image Processing*, vol. 16, no. 12, pp. 3005–3019, 2007.

[55] R. Jirik and T. Taxt, "Two dimensional blind Bayesian deconvolution of medical ultrasound images," *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, vol. 55, no. 10, pp. 2140–2153, 2008.

[56] C. Yu, C. Zhang, and L. Xie, "A blind deconvolution approach to ultrasound imaging," *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, vol. 59, no. 2, pp. 271–280, 2012.

[57] O. V. Michailovich, "Non-stationary blind deconvolution of medical ultrasound scans," in *Proceedings of SPIE: The International Society for Optical Engineering*, vol. 101391C, Mar. 2017.

[58] L. Roquette, M. M. J.-A. Simeoni, P. Hurley, and A. G. J. Besson, "On an analytical, spatially-varying, point-spread-function," in *IEEE International Ultrasound Symposium (IUS)*, Sep. 2017, Washington D.C., USA.

[59] J. G. Nagy and D. P. O'Leary, "Restoring images degraded by spatially variant blur," *SIAM Journal on Scientific Computing*, vol. 19, no. 4, pp. 1063–1082, Jul. 1998.

[60] Z. Chen, A. Basarab, and D. Kouamé, "Compressive deconvolution in medical ultrasound imaging," vol. 35, no. 3, pp. 728 – 737, 2016.

[61] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 3rd ed. Pearson Prentice Hall, Upper Saddle River, NJ, USA, 2008.

[62] B. P. Lathi, *Signal Processing and Linear Systems*. Oxford University Press, New York, NY, USA, 1998.

[63] G. S. Alberti, H. Ammari, F. Romero, and T. Wintz, "Mathematical analysis of ultrafast ultrasound imaging," *SIAM Journal on Applied Mathematics*, no. 77, pp. 1 – 25, 2017.

[64] M. I. Florea, A. Basarab, D. Kouamé, and S. A. Vorobyov, "An axially variant kernel imaging model applied to ultrasound image reconstruction," *IEEE Signal Processing Letters*, vol. 25, no. 7, pp. 961–965, Jul. 2018.

[65] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 67, no. 2, pp. 301–320, 2005.

[66] C. Quinsac, A. Basarab, and D. Kouamé, "Frequency domain compressive sampling for ultrasound imaging," *Advances in Acoustics and Vibration*, vol. 12, pp. 1–16, 2012.

[67] T. Szasz, A. Basarab, M.-F. Vaida, and D. Kouamé, "Elastic-net based beamforming in medical ultrasound imaging (regular paper)," Apr. 2016, Prague, Czech Republic, pp. 477–480.

[68] B. Byram, "Aperture domain model image reconstruction (ADMIRE) for improved ultrasound imaging," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, Mar. 2017, pp. 6250 –6253.

[69] J. A. Jensen and N. B. Svendsen, "Calculation of pressure fields from arbitrarily shaped, apodized, and excited ultrasound transducers," *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, vol. 39, no. 2, pp. 262–267, Mar. 1992.

[70] J. A. Jensen, "Field: A program for simulating ultrasound systems," *Medical and Biological Engineering and Computing*, vol. 34, pp. 351–353, 1996.

[71] M. J. Ackerman, "The visible human project," *Proceedings of the IEEE*, vol. 86, no. 3, pp. 504–511, 1998.

Publication I

Mihai I. Florea and Sergiy A. Vorobyov. A Robust FISTA-like Algorithm. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, New Orleans, USA, pp. 4521–4525, Mar. 2017.

A ROBUST FISTA-LIKE ALGORITHM

Mihai I. Florea and Sergiy A. Vorobyov

Department of Signal Processing and Acoustics, Aalto University, Espoo, Finland

ABSTRACT

The Fast Iterative Shrinkage-Thresholding Algorithm (FISTA) is regarded as the state-of-the-art among a number of proximal gradient-based methods used for addressing large-scale optimization problems with simple but non-differentiable objective functions. However, the efficiency of FISTA in a wide range of applications is hampered by a simple drawback in the line search scheme. The local estimate of the Lipschitz constant, the inverse of which gives the step size, can only increase while the algorithm is running. As a result, FISTA can slow down significantly if the initial estimate of the Lipschitz constant is excessively large or if the local Lipschitz constant decreases in the vicinity of the optimal point. We propose a new FISTA-like method endowed with a robust step size search procedure and demonstrate its effectiveness by means of a rigorous theoretical convergence analysis and simulations.

Index Terms— FISTA, backtracking, line search, convergence

1. INTRODUCTION

Simple continuous convex optimization problems are used to model many inverse problems and several simple classification tasks, particularly in imaging applications. Often, as in the case of sparse inverse problems, the objective is not differentiable in certain parts of the search space [1]. Accelerated algorithms that rely on gradient information (e.g. [2]) cannot be used directly to solve such problems. However, if the problem objective has a composite structure, certain algorithms are effective when supplied with *proximal gradient* information, instead of gradient information [3]. Although the number of variables can be large, usually of the order of millions [1], with recent advances in graphics processors it is possible to compute the proximal gradient on a single machine without the need for expensive communication between processing nodes. Consequently, proximal gradient methods are increasingly employed for addressing composite problems, and have become the subject of very active research [4–8]. Among proximal gradient methods, the Fast Iterative Shrinkage-Thresholding Algorithm (FISTA) [9] is currently regarded as the state-of-the-art [1, 10].

FISTA is designed to solve the following problem:

$$\min_{\boldsymbol{x}\in Q} F(\boldsymbol{x}) = f(\boldsymbol{x}) + \Psi(\boldsymbol{x}), \tag{1}$$

where $Q \subseteq \mathbb{R}^n$ is a closed convex set, $\boldsymbol{x}$ is a vector of optimization variables, and F is the objective function, which has a composite structure, i.e., f is a convex differentiable function with Lipschitz continuous gradient (Lipschitz constant L_f) and Ψ is a convex function that may not be differentiable nor defined outside Q. However, Ψ is simple in the sense that the proximal operator

$$\text{prox}_{\tau\Psi}(\boldsymbol{v}) = \arg\min_{\boldsymbol{x}\in Q} \left(\Psi(\boldsymbol{x}) + \frac{1}{2\tau}\|\boldsymbol{x}-\boldsymbol{v}\|_2^2 \right) \tag{2}$$

can be computed for any v in the search space and any non-negative step size τ in $\mathcal{O}(n)$ time. The popularity of FISTA owes to its simplicity and speed, but mostly to its generality [10]. The algorithm is unaware of the nature of the objective function or the problem constraints. It progresses using a sequence of calls to a proximal gradient routine, each involving one call to the proximal operator.

When the Lipschitz constant L_f is not known in advance, FISTA employs a simple backtracking "line search" procedure. However, the effectiveness of FISTA is hampered by a simple drawback in the search scheme, namely the estimate of the Lipschitz constant *can only increase* while the algorithm is running. As the step size is set to be the inverse of the Lipschitz constant estimate, the line search may slow the progress of the algorithm. In two situations, the drawbacks of the search are evident: (i) L_0, the initial estimate of L_f, far exceeds the actual value and (ii) the local curvature of f is large in the vicinity of the initial iterates but decreases around the optimal point.

A number of approaches addressing both aforementioned issues have been proposed in the literature. A method that predates FISTA, which we choose to call Nesterov's Accelerated Multistep Gradient Scheme (Nesterov's AMGS) [11], does feature a step size increase schedule. While having comparable theoretical convergence guarantees to FISTA, it is slower in the most common applications [12]. A variation of Nesterov's AMGS more similar to FISTA, mentioned in [3], is nominally equipped with a step size increase option when "conditions permit". However, the study does not quantitatively describe these conditions nor does it provide any theoretical convergence guarantees. In a recent study [13], FISTA has been equipped with an "exact" line search procedure. This version comes with a rigorous convergence analysis but assumes that the objective function is known to the algorithm, detracting from the generality of FISTA's black-box philosophy.

In this work, we propose an algorithm that alleviates the drawbacks of FISTA in the above mentioned situations, rendering it *robust* in the sense that it can be applied without parameter adjustment to the full spectrum of problems it addresses. Furthermore, our method does not restrict the generality of FISTA and does not alter the theoretical convergence guarantees while surpassing FISTA in practice. We support our findings with simulation results.

2. ROBUST LINE SEARCH FISTA

Our goal is to create an algorithm that can dynamically adjust the Lipschitz constant estimate at every iteration. The simplest and most straightforward way of achieving this is by decreasing the Lipschitz constant estimate slightly at the beginning of every iteration, relying on backtracking to correct excessive reduction. This search strategy is not applicable to FISTA (i.e., convergence cannot be theoretically guaranteed) because this method collects insufficient information while it is running. In FISTA, the first iteration $k = 0$ is a proximal point step [10]. At every subsequent iteration $k \geq 1$,

the new iterate $\boldsymbol{x}_{k+1}$ (estimate of optimal point $\boldsymbol{x}^*$) is obtained by querying the proximal gradient not at the previous iterate $\boldsymbol{x}_k$, but at a point $\boldsymbol{y}_{k+1}$ obtained from the two preceding iterates $\boldsymbol{x}_k$ and $\boldsymbol{x}_{k-1}$ through extrapolation, the extent of which depends on terms t_k and t_{k+1} of a recursively defined weight sequence $\{t_i\}_{i\geq 1}$. Although not maintained explicitly while running, FISTA's convergence analysis relies on an auxiliary sequence (which we denote by $\{\boldsymbol{z}_i\}_{i\geq 0}$) that is updated in parallel with $\{\boldsymbol{x}_i\}_{i\geq 0}$. Note that $\boldsymbol{y}_{k+1}$ can be obtained as a convex combination of $\boldsymbol{x}_k$ and $\boldsymbol{z}_k$, with the weighting determined by t_k and t_{k+1}.

FISTA benefits from the same simplification employed by the Fast Gradient Method (FGM) [2]. When the step size is non-increasing, the sequence $\{t_i\}_{i\geq 1}$ can be determined a priori, irrespective of Lipschitz constant estimate values. Maintaining at every iteration k the accumulated weight property

$$T_{k+1} = \sum_{i=1}^{k+1} t_i = t_{k+1}^2, \quad \forall k \geq 0, \tag{3}$$

guarantees an $\mathcal{O}(\frac{1}{k^2})$ rate of convergence, optimal for a class of first-order algorithms introduced in [2]. By relaxing the non-increasing step size assumption, the weight sequence $\{t_i\}_{i\geq 1}$ can be updated to take into account the current and past Lipschitz constant estimates, yielding a more robust algorithm that retains the $\mathcal{O}(\frac{1}{k^2})$ rate of convergence. Specifically, we can replace the accumulated weight property (3) with

$$T_{k+1} = \sum_{i=1}^{k+1} \frac{t_i}{L_i} \geq L_{k+1} t_{k+1}^2, \quad \forall k \geq 0, \tag{4}$$

where L_{k+1} represents the new Lipschitz constant estimate obtained at iteration k. Equality in (4) ensures the fastest theoretical convergence rate but our framework accommodates also methods that violate equality to trade off speed for other desirable properties, such as weak convergence of iterates [7] or ease of interpretation [14].

Using equality in (4), we propose the method outlined in Algorithm 1. Our notation differs slightly from the one used by FISTA [9]. We define the quadratic function $U_{L,y}(\boldsymbol{x})$ as

$$U_{L,y}(\boldsymbol{x}) = f(\boldsymbol{y}) + \langle \nabla f(\boldsymbol{y}), \boldsymbol{x} - \boldsymbol{y} \rangle + \frac{L}{2}\|\boldsymbol{x} - \boldsymbol{y}\|_2^2. \tag{5}$$

Then, by fixing the step size in (2) to be $\tau = 1/L$, the proximal gradient expression becomes

$$p_L(\boldsymbol{y}) = \arg\min_{\boldsymbol{x}\in Q} \left(U_{L,y}(\boldsymbol{x}) + \Psi(\boldsymbol{x})\right) \tag{6}$$

$$= \text{prox}_{\tau\Psi}\left(\boldsymbol{y} - \tau \bigtriangledown f(\boldsymbol{y})\right). \tag{7}$$

While the formulation of our algorithm appears greatly dissimilar to FISTA, in fact, we can obtain an algorithm that is *completely equivalent* to FISTA (which we designate as z-FISTA) by simply removing line 4 and by modifying lines 6 and 19. For z-FISTA, line 6 reads instead as

$$\hat{t} := \frac{1 + \sqrt{1 + 4T_k}}{2} \tag{8}$$

and line 19 is replaced with $\boldsymbol{z}_{k+1} = \boldsymbol{z}_k + \hat{t}(\hat{\boldsymbol{x}} - \hat{\boldsymbol{y}})$.

The relation of the remainder of parameters in Algorithm 1 to those present in z-FISTA is listed in Table 1. In the proposed method, the accumulated weight T_{k+1} constitutes a valid convergence rate at each iteration k (see Section 3 for proof). The quantity T_{k+1} does not hold the same meaning in z-FISTA, where a valid convergence rate is instead given by $\frac{T_{k+1}}{L_{k+1}}$. This is poorer than in our method. In FISTA, $\{\boldsymbol{z}_i\}_{i\geq 0}$ and $\{T_i\}_{i\geq 0}$ are abstracted away. Similarly, in the proposed method, there is no need to maintain $\{\boldsymbol{y}_i\}_{i\geq 1}$ nor $\{t_i\}_{i\geq 1}$ across iterations. Instead, we update only current estimates of these sequences.

Algorithm 1 A robust FISTA-like algorithm

1: $\boldsymbol{z}_0 = \boldsymbol{x}_0$
2: $T_0 = 0$
3: **for** k = 0,...,K-1 **do**
4: $\quad \hat{L} := \gamma_d L_k$
5: $\quad$ **loop**
6: $\quad\quad \hat{t} := \frac{1+\sqrt{1+4\hat{L}T_k}}{2\hat{L}}$
7: $\quad\quad \hat{T} := T_k + \hat{t}$
8: $\quad\quad \hat{\boldsymbol{y}} := \frac{1}{\hat{T}}(T_k\boldsymbol{x}_k + \hat{t}\boldsymbol{z}_k)$
9: $\quad\quad \hat{\boldsymbol{x}} := p_{\hat{L}}(\hat{\boldsymbol{y}})$
10: $\quad\quad$ **if** $f(\hat{\boldsymbol{x}}) \leq U_{\hat{L},\hat{\boldsymbol{y}}}(\hat{\boldsymbol{x}})$ **then**
11: $\quad\quad\quad$ Break from loop
12: $\quad\quad$ **else**
13: $\quad\quad\quad \hat{L} := \gamma_u \hat{L}$
14: $\quad\quad$ **end if**
15: $\quad$ **end loop**
16: $\quad L_{k+1} = \hat{L}$
17: $\quad \boldsymbol{x}_{k+1} = \hat{\boldsymbol{x}}$
18: $\quad T_{k+1} = \hat{T}$
19: $\quad \boldsymbol{z}_{k+1} = \boldsymbol{z}_k + \hat{t}\hat{L}(\hat{\boldsymbol{x}} - \hat{\boldsymbol{y}})$
20: **end for**

Table 1. Parameters and variables used by our method

Type	Symbol	Domain	Description	z-FISTA equivalent
Input	$\boldsymbol{x}_0$	$\mathbb{R}^n$	initial estimate of $\boldsymbol{x}^*$	same
Input	L_0	$(0,\infty)$	initial estimate of L_f	same
Input	γ_u	$(1,\infty)$	increase rate of $\hat{L}$	same
Input	γ_d	$(0,1)$	decrease rate of $\hat{L}$	none
Internal	$\hat{L}$	$(0,\infty)$	estimate of L_f	same
Internal	$\hat{\boldsymbol{x}}$	Q	estimate of $\boldsymbol{x}_{k+1}$	same
Internal	$\hat{\boldsymbol{y}}$	$\mathbb{R}^n$	estimate of $\boldsymbol{y}_{k+1}$	$\boldsymbol{y}_{k+1}$
Internal	$\hat{t}$	$(0,\infty)$	weight of $\boldsymbol{z}_{k+1}$	$\frac{t_{k+1}}{L_{k+1}}$
Internal	$\hat{T}$	$(0,\infty)$	estimate of T_{k+1}	$\frac{T_{k+1}}{L_{k+1}}$
Output	$\boldsymbol{x}_K$	Q	final estimate of $\boldsymbol{x}^*$	same

Our method cannot be benchmarked directly against FISTA and Nesterov's AMGS in terms of theoretical computational complexity. Each method calls a dynamic mix of functions which, depending on the problem specification, may have vastly varying relative complexities. Table 2 provides a detailed description of the type and number of function calls in several stages of an iteration. Our method requires more computation than FISTA for a backtracking operation while showing no increase in complexity when no backtracks occur. Nesterov's AMGS cannot be compared in any algorithmic state as it was designed to work without the need to implement function value calls to f. An iteration of Nesterov's AMGS does, however, require at least two projection calls (proximal operator plus gradient computation) which gives it a clear disadvantage when these operations are more complex than calls to f. Overall, our method strikes a balance in a variety of situations, further contributing to its robustness.

Table 2. Per iteration complexity, measured in terms of operator calls, of FISTA, Nesterov's AMGS, and our method

	FISTA			Nesterov's AMGS			Our method		
	f	∇f	$\text{prox}_{\tau\Psi}$	f	∇f	$\text{prox}_{\tau\Psi}$	f	∇f	$\text{prox}_{\tau\Psi}$
Step size validation (lines 4 to 11)	2	1	1	0	2	1	2	1	1
Backtrack (lines 4 to 15)	1	0	1	0	2	1	2	1	1
State update (lines 16 to 19)	0	0	0	0	0	1	0	0	0
Iteration without backtrack (lines 4 to 19)	2	1	1	0	2	2	2	1	1

3. CONVERGENCE ANALYSIS

While our method does not keep track of the sequences $\{\boldsymbol{y}_k\}_{k\geq 1}$ and $\{t_k\}_{k\geq 1}$, they can be easily recovered. Lines 6, 7, 8 and 18 in Algorithm 1 imply that

$$t_{k+1} = T_{k+1} - T_k, \quad \forall k \geq 0, \tag{9}$$
$$T_{k+1}\boldsymbol{y}_{k+1} = T_k\boldsymbol{x}_k + t_{k+1}\boldsymbol{z}_k, \quad \forall k \geq 0. \tag{10}$$

Given that for every $k \geq 1$, $\boldsymbol{x}_k$ satisfies relations

$$f(\boldsymbol{x}_k) \leq U_{L_k,\boldsymbol{y}_k}(\boldsymbol{x}_k), \tag{11}$$
$$\boldsymbol{x}_k = p_{L_k}(\boldsymbol{y}_k), \tag{12}$$

enforced by lines 9, 10, 16 and 17 in Algorithm 1, it follows that [9, Lemma 2.3] holds for all $k \geq 1$ and $\boldsymbol{x} \in \mathbb{R}^n$, that is

$$F(\boldsymbol{x}) - F(\boldsymbol{x}_k) \geq \frac{L_k}{2}\|\boldsymbol{x}_k - \boldsymbol{y}_k\|_2^2 + L_k\langle \boldsymbol{y}_k - \boldsymbol{x}, \boldsymbol{x}_k - \boldsymbol{y}_k\rangle. \tag{13}$$

Let us consider the sequence $\{\Delta_k\}_{k\geq 0}$, defined as

$$\Delta_k = T_k(F(\boldsymbol{x}_k) - F^*) + \frac{1}{2}\|\boldsymbol{z}_k - \boldsymbol{x}^*\|_2^2, \quad \forall k \geq 0. \tag{14}$$

We aim to prove that this sequence is non-increasing. Indeed, applying (13) at iteration $k+1$ (for all $k \geq 0$) using $\boldsymbol{x}_k$ and $\boldsymbol{x}^*$ as values of $\boldsymbol{x}$ we obtain

$$F(\boldsymbol{x}_k) - F(\boldsymbol{x}_{k+1}) \geq \frac{L_{k+1}}{2}\|\boldsymbol{x}_{k+1} - \boldsymbol{y}_{k+1}\|_2^2 + L_{k+1}\langle \boldsymbol{y}_{k+1} - \boldsymbol{x}_k, \boldsymbol{x}_{k+1} - \boldsymbol{y}_{k+1}\rangle, \tag{15}$$
$$F(\boldsymbol{x}^*) - F(\boldsymbol{x}_{k+1}) \geq \frac{L_{k+1}}{2}\|\boldsymbol{x}_{k+1} - \boldsymbol{y}_{k+1}\|_2^2 + L_{k+1}\langle \boldsymbol{y}_{k+1} - \boldsymbol{x}^*, \boldsymbol{x}_{k+1} - \boldsymbol{y}_{k+1}\rangle. \tag{16}$$

Lines 6, 7, 16 and 18 ensure that (4) holds with equality. Using (9), (10), and (4) in $T_k \cdot (15) + t_{k+1} \cdot (16)$ we obtain

$$T_k(F(\boldsymbol{x}_k) - F^*) - T_{k+1}(F(\boldsymbol{x}_{k+1}) - F^*) \geq L_{k+1}^2 t_{k+1}^2\|\boldsymbol{x}_{k+1} - \boldsymbol{y}_{k+1}\|_2^2 + t_{k+1}L_{k+1}\langle \boldsymbol{z}_k - \boldsymbol{x}^*, \boldsymbol{x}_{k+1} - \boldsymbol{y}_{k+1}\rangle. \tag{17}$$

Lines 16 and 19 translate into the following recursion rule:

$$\boldsymbol{z}_{k+1} = \boldsymbol{z}_k + t_{k+1}L_{k+1}(\boldsymbol{x}_{k+1} - \boldsymbol{y}_{k+1}), \quad \forall k \geq 0. \tag{18}$$

Then, using (18) in (17) and rearranging terms, we obtain the desired result,

$$\Delta_{k+1} \leq \Delta_k, \quad \forall k \geq 0. \tag{19}$$

This last inequality implies that every term Δ_k, $k \geq 1$, is upper bounded by Δ_0. Given that $\Delta_0 = \frac{1}{2}\|\boldsymbol{x}_0 - \boldsymbol{x}^*\|_2^2$ and that the quantity $\frac{1}{2}\|\boldsymbol{z}_k - \boldsymbol{x}^*\|_2^2$ is always non-negative, we can write down a convergence rate explicitly as

$$F(\boldsymbol{x}_k) - F^* \leq \frac{1}{2T_k}\|\boldsymbol{x}_0 - \boldsymbol{x}^*\|_2^2, \quad \forall k \geq 1. \tag{20}$$

Clearly, T_k constitutes a valid convergence rate. To obtain a simple closed form convergence rate, it suffices to find a simple lower bound for T_k. When $L_k \geq L_f$, due to the Lipschitz continuous property of ∇f, inequality (11) holds regardless of the values of $\boldsymbol{x}_k$ and $\boldsymbol{y}_k$. Hence, the backtracking search will never produce a value of L_k beyond $\gamma_u L_f$. Therefore, combining (4) with $L_k \leq \gamma_u L_f$ we obtain

$$T_{k+1} \geq T_k + \frac{1}{2\gamma_u L_f} + \sqrt{\frac{1}{4(\gamma_u L_f)^2} + \frac{T_k}{\gamma_u L_f}}, \quad \forall k \geq 0. \tag{21}$$

Using (21) and $T_0 = 0$, it follows through induction that

$$T_k \geq \frac{(k+1)^2}{4\gamma_u L_f}, \quad \forall k \geq 1. \tag{22}$$

Finally, substituting the lower bound on T_k (22) in (20) we obtain the same quadratic convergence rate as the original FISTA [9, Theorem 4.4], namely

$$F(\boldsymbol{x}_k) - F^* \leq \frac{2\gamma_u L_f}{(k+1)^2}\|\boldsymbol{x}_0 - \boldsymbol{x}^*\|_2^2, \quad \forall k \geq 1. \tag{23}$$

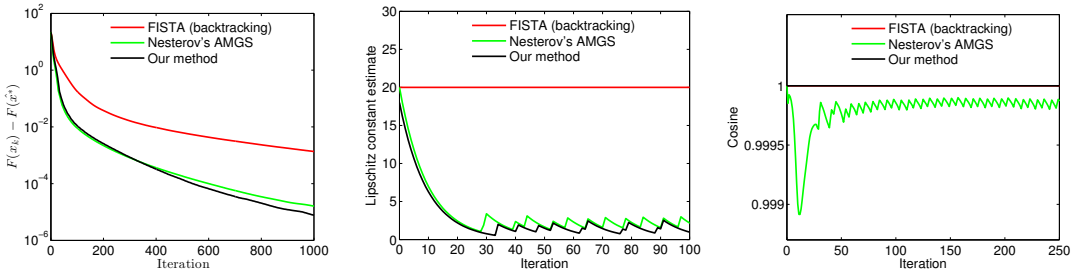
Note that our convergence analysis is more general than the one provided in [9]. Inequality (19) applies to FISTA as well (by replacing T_k with $\bar{T}_k = t_k^2/L_k$ for $k \geq 1$ and setting $T_0 = 0$) and can be used to obtain the same convergence rate for variations on the weight update rule (8).

4. NUMERICAL ANALYSIS

The performance of our method (Algorithm 1) was tested and compared to that of FISTA with backtracking line search and Nesterov's AMGS on the l_1 regularized deblurring of a simple test image. For ease of benchmarking, we used the experimental setup from [9, Section 5.1]. The composite objective function is given by

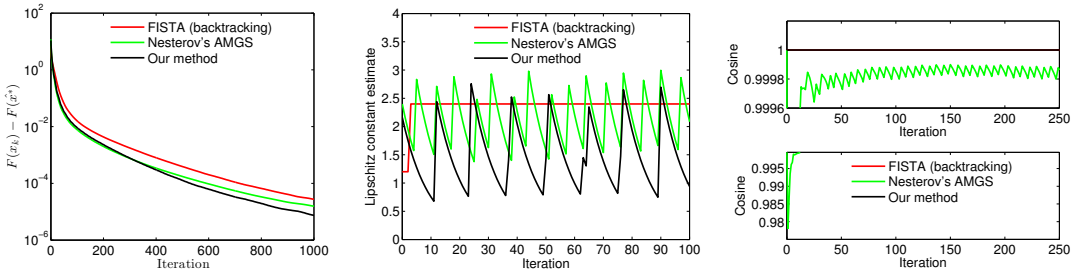
$$f(\boldsymbol{x}) = \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2, \qquad \Psi(\boldsymbol{x}) = \lambda\|\boldsymbol{x}\|_1, \tag{24}$$

where $\boldsymbol{A} = \boldsymbol{R}\boldsymbol{W}$; $\boldsymbol{R}$ is a matrix representing Gaussian blur (9×9 pixel kernel, standard deviation 4.0, reflexive boundary conditions [15]); $\boldsymbol{W}$ is the inverse three-stage Haar wavelet transform; $\boldsymbol{b}$ is obtained by applying $\boldsymbol{R}$ to the 256×256 cameraman test image (pixel values scaled to the $[0, 1]$ range), followed by the addition of Gaussian noise (zero-mean, standard deviation 10^{-3}). Here, ∇f has a Lipschitz constant value $L_f = 2.0$, computed as the maximum eigenvalue of a symmetric Toeplitz-plus-Hankel matrix, according to [15], and $\lambda = 2 \cdot 10^{-5}$ is a regularization parameter. In addition, we chose $\gamma_u = 2.0$ and $\gamma_d = 0.9$ for each method tested, as these values were suggested in [3] to "provide good performance in many applications". Two scenarios are considered: a pathologically overestimated initial guess $L_0 = 10L_f$ (Fig. 1) and a normally underestimated $L_0 = 0.3L_f$ (Fig. 2).



(a) Convergence rate (b) Lipschitz constant estimate (c) Inertial degree

Fig. 1. Comparison of FISTA, Nesterov's AMGS, and our method for an overestimated initial Lipschitz constant: $L_0 = 10L_f$



(a) Convergence rate (b) Lipschitz constant estimate (c) Inertial degree

Fig. 2. Comparison of FISTA, Nesterov's AMGS, and our method for an underestimated initial Lipschitz constant: $L_0 = 0.3L_f$

Convergence is measured in terms of the difference between objective function values and an optimal value estimate, during the first 1000 iterations (Figs. 1(a) and 2(a)). This estimate was computed as $F(\hat{\boldsymbol{x}}^*)$, where $\hat{\boldsymbol{x}}^*$ is the iterate obtained after running fixed step size FISTA with the correct Lipschitz constant parameter for 10000 iterations. Key algorithm state parameters, such as Lipschitz constant estimates (Figs. 1(b) and 2(b)) and inertial degrees (Figs. 1(c) and 2(c)) are shown only during the first 100 iterations, as subsequent iterations did not reveal more information. Inertial degrees are defined at every iteration $k \geq 0$ as the cosine of the angle between $\boldsymbol{y}_{k+1}$ and $\boldsymbol{x}_k$ at $\boldsymbol{x}_{k-1}$. When two of these points match, the inertial degree is set to 1.

In both scenarios, after the first 400 iterations, our method clearly surpasses the others in terms of function value (Figs. 1(a) and 2(a)). FISTA converges slowly, especially in the pathological case where, as expected, FISTA is unable to reduce its Lipschitz constant estimate (Fig. 1(b)), whereas the other methods are able to decrease their estimates at comparable rates during the first 30 iterations. Under normal conditions (Fig. 2(b)), FISTA quickly increases its estimate in the first iterations after which the value reaches a saturation level. The other methods are constantly adjusting their estimates. In both situations, our method produces on average a lower L_k than the other two methods. FISTA's inability to reduce L_k accounts for its high estimates whereas Nesterov's AMGS has a stricter backtracking condition than our method or FISTA, leading to more backtracks.

In our method, just as in FISTA, $\boldsymbol{y}_{k+1}$, $\boldsymbol{x}_k$ and $\boldsymbol{x}_{k-1}$ are collinear (Figs. 1(c) and 2(c)). However, in Nesterov's AMGS they are not, contradicting the notion found in several monographs in the field (e.g. [10, 16]) that all accelerated first order methods rely on extrapolation. We provide in [12] a rigorous proof of collinearity in the proposed method and corroborate the superiority of our algorithm with a more detailed performance analysis.

In summary, our method can be regarded as a hybrid of FISTA with its collinear iterates (Figs. 1(c) and 2(c)) and Nesterov's AMGS with its dynamic step search procedure (Figs. 1(b) and 2(b)), benefiting from the strengths of these methods while alleviating the drawbacks. Namely, our method produces more accurate estimates of the local curvature of f (unlike the artificially high estimates of FISTA) and is able to utilize both gradient and subgradient information (whereas Nesterov's AMGS updates a weighted average of gradients without taking into consideration the subgradient of Ψ), resulting in larger steps and, consequently, faster convergence.

5. CONCLUSION

By updating the weight sequence to take into account the current and past Lipschitz constant estimates, we have devised a FISTA-like algorithm with a robust step size search strategy. We have shown that the same theoretical convergence rate of $\mathcal{O}(\frac{1}{k^2})$ applies to our method, with a provably smaller constant. Simulation results on the very problem FISTA was introduced to solve show that our method surpasses both FISTA and the more complex Nesterov's AMGS, without the need to adjust any parameters.

The properties of the proposed method follow naturally from the augmented estimate sequence framework [12]. In fact, our method is a particular case of the Accelerated Composite Gradient Method, a general-purpose optimization scheme [12]. Thus, the concepts presented in this work are of importance to the entire field of accelerated optimization algorithms.

6. REFERENCES

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

[2] Y. Nesterov, *Introductory lectures on convex optimization. Applied optimization, vol. 87*. Kluwer Academic Publishers, Boston, 2004.

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

[4] A. Auslender and M. Teboulle, "Interior gradient and proximal methods for convex and conic optimization," *SIAM J. Optim.*, vol. 16, no. 3, pp. 697–725, 2006.

[5] G. Lan, Z. Lu, and R. D. Monteiro, "Primal-dual first-order methods with $\mathcal{O}(1/\epsilon)$ iteration-complexity for cone programming," *Math. Program.*, vol. 126, no. 1, pp. 1–29, 2011.

[6] P. Tseng, "On accelerated proximal gradient methods for convex-concave optimization," May 2008, submitted to SIAM J. Optim. Available: http://www.mit.edu/~dimitrib/PTseng/papers/apgm.pdf

[7] 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, May 2015.

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

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

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

[11] Y. Nesterov, "Gradient methods for minimizing composite objective function," CORE, Université Catholique de Louvain, Tech. Rep. 76, Sep. 2007.

[12] M. I. Florea and S. A. Vorobyov, "An accelerated composite gradient method for large-scale composite objective problems," *arXiv preprint* arXiv:1612.02352 [math.OC], Dec. 2016.

[13] Z. Zhang and V. Saligrama, "RAPID: Rapidly accelerated proximal gradient algorithms for convex minimization," *Proc. IEEE Intern. Conf. Acoust., Speech and Signal Process. (ICASSP)*, Brisbane, Australia, Apr. 2015, pp. 3796–3800.

[14] W. Su, S. Boyd, and E. Candes, "A differential equation for modeling Nesterov's accelerated gradient method: Theory and insights," *Advances in Neural Inform. Process. Syst. 27*, Montréal, Canada, Dec. 2014, pp. 2510–2518.

[15] P. Hansen, J. Nagy, and D. O'Leary, *Deblurring images*. SIAM, Philadelphia, 2006.

[16] D. P. Bertsekas, *Convex optimization algorithms*. Athena Scientific, Belmont, 2015.

Publication II

Mihai I. Florea and Sergiy A. Vorobyov. An Accelerated Composite Gradient Method for Large-scale Composite Objective Problems. Accepted for publication in *IEEE Transactions on Signal Processing*, May 2018.

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 inverse problems, tabulated function minimization problems, and machine learning tasks can be expressed as large-scale optimization problems with a composite objective structure, where the Lipschitz constant of the smooth part gradient is not known. For other problems, as in l_1-regularized logistic regression, the global Lipschitz constant is known but its local values may only be a fraction of the global value. In all the above cases, the smooth part may be strongly convex as well. Numerous methods have been proposed to deal with different instances within this class of problems. However, these do not account for the characteristics of the entire problem class, leading to performance degradation or outright divergence outside their scope. The most generic among them are black-box accelerated first-order methods, related to either Nesterov's Fast Gradient Method (FGM) or the Accelerated Multistep Gradient Scheme (AMGS), which were developed and analyzed using the estimate sequence mathematical framework. In this work, we introduce 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 the entire composite problem class. Moreover, ACGM is endowed with an efficient dynamic Lipschitz constant estimation (line-search) procedure. Motivated by the absence of a reliable complexity measure applicable to all first-order methods, we also introduce the wall-clock time unit (WTU). The WTU accounts for variations in algorithmic per-iteration complexity and more consistently reflects algorithm 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 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 that have 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

M. I. Florea (E-mail: mihai.florea@aalto.fi) and S. A. Vorobyov (E-mail: sergiy.vorobyov@aalto.fi) are with Aalto University, Department of Signal Processing and Acoustics, FI-00076, AALTO, Finland. This work has been partially supported by the Academy of Finland (Grant No. 299243). A particular case of the algorithm introduced in this paper was presented at the 42nd IEEE ICASSP, New Orleans, USA, March 2017.

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 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 objective structure, 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

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

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.

For a given function Ψ, we also define the set of composite parabolae $\mathcal{P}_\Psi \triangleq \{\psi + c\Psi \mid c\geq 0,\ \psi\in\mathcal{P}\}$.

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 $A_kF(\boldsymbol{x}_k)$. 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

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

[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]).

The estimate sequence property in (8) is more stringent than the provable convergence rate expression (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 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**

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 $u_{k+1}(\boldsymbol{x})$, a local upper bound on F (Algorithm 1, line 7). Note that $u_{k+1}(\boldsymbol{x})$ is not related to ψ_k^*. The estimate function ψ_k is incremented with a global lower bound $w_{k+1}(\boldsymbol{x})$ 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

$$W_k(\boldsymbol{x}) = \frac{1}{A_k}\sum_{i=1}^{k} a_i w_i(\boldsymbol{x}), \quad k \geq 1.$$

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 $w_{k+1}(\boldsymbol{x})$ and upper bounds $u_{k+1}(\boldsymbol{x})$ 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 $k \geq 0$. 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

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

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.

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 $L_{k+1} a^2 = 2(A_k + a)(1 + \mu A_k)$
$\mathcal{F}_{\boldsymbol{y}}(\boldsymbol{x}_k, \psi_k, A_k, a_{k+1})$	$\frac{A_k \gamma_{k+1} \boldsymbol{x}_k + a_{k+1} \gamma_k \boldsymbol{v}_k}{A_k \gamma_{k+1} + a_{k+1} \gamma_k}$	$\frac{A_k \boldsymbol{x}_k + a_{k+1} \boldsymbol{v}_k}{A_k + a_{k+1}}$

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

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

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

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

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

is equivalent to that of f', stated as

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

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 $k \geq 0$, 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* $k \geq 0$*, then the objective* F *is lower bounded as*

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

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* $\{\psi'_k(\boldsymbol{x})\}_{k\geq 0}$ which we define, using the notation and conventions from Subsection II-A, as

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

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

$$A_k F(\boldsymbol{x}_k) \leq \psi'^*_k, \quad k \geq 0, \tag{20}$$

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

$$A_k F(\boldsymbol{x}_k) \leq \psi'^*_k = \psi^*_k + A_k(F(\boldsymbol{x}^*) - W_k(\boldsymbol{x}^*))$$
$$\leq \psi^*_k + H_k - \psi_k(\boldsymbol{x}^*) \leq H_k, \quad k \geq 0.$$

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

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

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

$$\Gamma_{k+1} \leq \Gamma_k, \quad k \geq 0. \tag{21}$$

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:

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

$$\psi_k'(\boldsymbol{x}) = \psi_k'^* + \frac{\gamma_k}{2}\|\boldsymbol{x} - \boldsymbol{v_k}\|_2^2. \tag{23}$$

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

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

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 simplest way to ensure $\mathcal{A}_{k+1} \geq 0$ is by maintaining

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

The vector terms in $\mathcal{B}_{k+1}$ may form an obtuse angle so we set $\boldsymbol{s}_{k+1} = 0$, which 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 ensuring that γ_{k+1} is as large as possible by setting $\gamma_0 = 1$. 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

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

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

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

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

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
    $\hat{a}_{k+1} := \frac{1}{2(\hat{L}_{k+1} - \mu_f)} \left(\gamma_k + A_k\mu + \sqrt{(\gamma_k + A_k\mu)^2 + 4(\hat{L}_{k+1} - \mu_f)A_k\gamma_k}\right)$
    $\hat{A}_{k+1} := A_k + \hat{a}_{k+1}$
    $\hat{\gamma}_{k+1} := \gamma_k + \hat{a}_{k+1}\mu$
    $\hat{\boldsymbol{y}}_{k+1} := \frac{1}{A_k\hat{\gamma}_{k+1} + \hat{a}_{k+1}\gamma_k}(A_k\hat{\gamma}_{k+1}\boldsymbol{x}_k + \hat{a}_{k+1}\gamma_k\boldsymbol{v}_k)$
    $\hat{\boldsymbol{x}}_{k+1} := \text{prox}_{\frac{1}{\hat{L}_{k+1}}\Psi}\left(\hat{\boldsymbol{y}}_{k+1} - \frac{1}{\hat{L}_{k+1}}\nabla f(\hat{\boldsymbol{y}}_{k+1})\right)$
    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
  $L_{k+1} := \hat{L}_{k+1},\ \boldsymbol{x}_{k+1} := \hat{\boldsymbol{x}}_{k+1}$
  $A_{k+1} := \hat{A}_{k+1},\ \gamma_{k+1} := \hat{\gamma}_{k+1}$
  $\boldsymbol{v}_{k+1} := \frac{1}{\hat{\gamma}_{k+1}}(\gamma_k\boldsymbol{v}_k + \hat{a}_{k+1}(\hat{L}_{k+1} + \mu_\Psi)\hat{\boldsymbol{x}}_{k+1} - \hat{a}_{k+1}(\hat{L}_{k+1} - \mu_f)\hat{\boldsymbol{y}}_{k+1})$
end for
return x_K

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 deals with worst case search behavior. To simplify notation, we introduce the local inverse condition number

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

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

$$L_{k+1} \leq L_u \triangleq \max\{r_u L_f, r_d L_0\}, \quad k \geq 0. \quad (33)$$

Let the worst case local inverse condition number be defined as

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

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

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

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

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

Proof: See Appendix E. ∎

F. ACGM in extrapolated form

An interesting property of FGM is that for all $k \geq 0$, 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

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

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, with the difference 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*

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

Proof: See Appendix F. ∎

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

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

where the auxiliary point extrapolation factor β_k is given by

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

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

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

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

$$\begin{aligned} &(L_{k+1}+\mu_\Psi)A_{k+1} - (L_{k+1}+\mu_\Psi)a_{k+1} \\ &\quad - \frac{L_{k+1}+\mu_\Psi}{L_k+\mu_\Psi}(L_k+\mu_\Psi)A_k = 0 \\ \Leftrightarrow\ &\gamma_{k+1}t_{k+1}^2 - \gamma_{k+1}t_{k+1} - \frac{L_{k+1}+\mu_\Psi}{L_k+\mu_\Psi}\gamma_k t_k^2 = 0 \\ \Leftrightarrow\ &t_{k+1}^2 + t_{k+1}(q_k t_k^2 - 1) - \frac{L_{k+1}+\mu_\Psi}{L_k+\mu_\Psi}t_k^2 = 0. \end{aligned} \tag{42}$$

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

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

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.

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

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

[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.

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 $\text{prox}_{\tau\Psi}(\boldsymbol{x})$ 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 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.

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

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 for 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 $r_u^{\text{AMGS}} = r_u^{\text{FISTA}} = 2.0$ and $r_d^{\text{AMGS}} = 0.9$ 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

[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	
		Computation	Iteration	Computation	Iteration	Computation	Iteration
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	
		Computation	Iteration	Computation	Iteration	Computation	Iteration
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

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 $L_0 = 0.3L_f$ (Figure 1) and a greatly overestimated $L_0 = 10L_f$. The convergence rate is measured as the difference between 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 (Figures 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 (Figures 1(b) and 1(e)), because of ACGM's low per-iteration complexity.

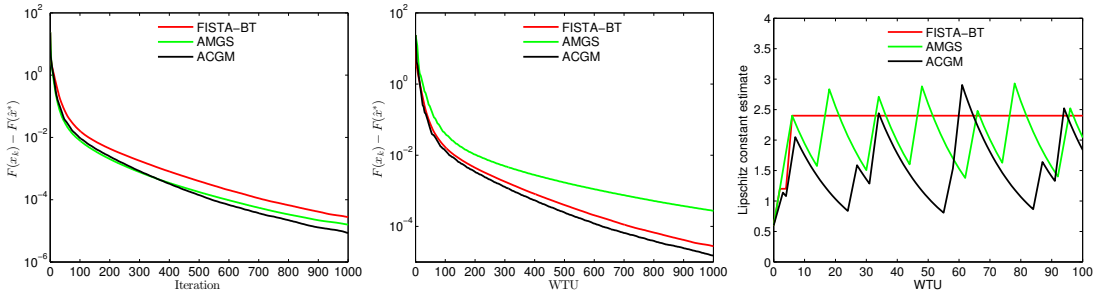
FISTA-BT lags behind in the overestimated case, regardless of the convergence measure (Figures 1(d) and 1(e)), and it is also slightly slower in the underestimated case (Figures 1(a) and 1(b)). The disadvantage of FISTA-BT lies in the inability of its line-search procedure to decrease the Lipschitz constant 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 Figures 1(c) and 1(f).

B. Logistic regression with elastic net

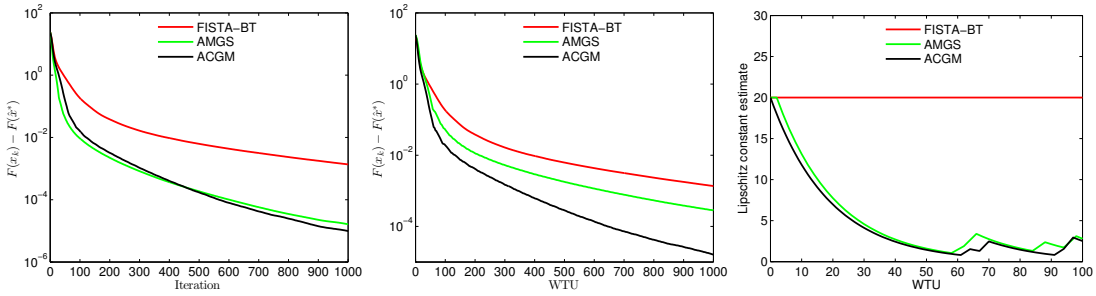
As a strongly convex application, we choose a randomly generated instance of the logistic regression classification task [29], regularized with an elastic net [30]. The objective function components are given by

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

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



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



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

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

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

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

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.

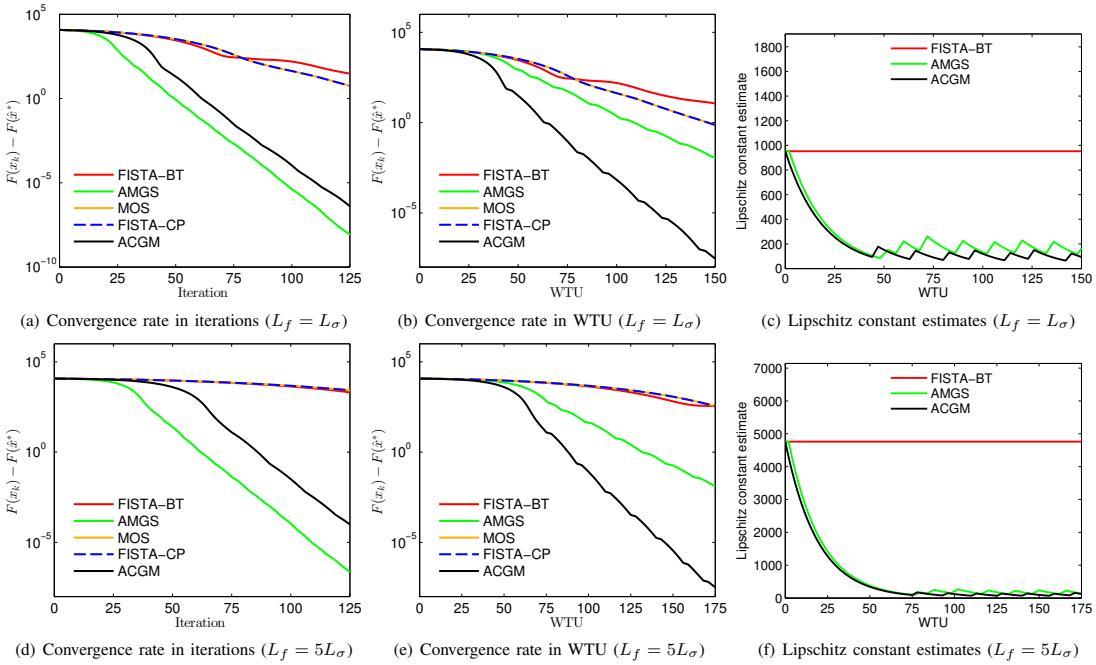
We benchmark ACGM against methods that have convergence guarantees. These methods are either equipped with a line-search procedure, such as FISTA and AMGS, or rely on L_f being known in advance, namely FISTA-CP and MOS. We do not include scAPG in our benchmark because $\mu_f = 0$. We also do not consider 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 $r_u^{\text{ACGM}} = r_u^{\text{AMGS}} = r_u^{\text{FISTA}} = 2.0$, $r_d^{\text{AMGS}} = 0.9$, and $r_d^{\text{ACGM}} = \sqrt{r_d^{\text{AMGS}}}$.

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

When indexing in iterations, AMGS converges the fastest (Figures 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 (Figure 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 (Figures 2(b) and 2(e)). Note that FISTA-CP and MOS display nearly identical convergence behaviors (Figures 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



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

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

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

considerably, even when an accurate value of L_f is supplied (Figures 2(a) and 2(b)). The reason is that the Lipschitz estimates of ACGM are several times smaller than the global value L_f (Figure 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 (Figure 2(f)). The estimates produced by the AMGS's damped relaxation condition are considerably higher than those of ACGM, further contributing for the ACGM's superior convergence behavior in WTU (Figures 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 $\tau^{\text{FISTA}-\text{CP}} = \frac{1}{L_f}$. 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 $r_u^{\text{ACGM}} = r_u^{\text{FISTA}}$, ACGM has superior theoretical convergence guarantees to FISTA, namely*

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

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

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

TABLE V
FEATURES OF BLACK-BOX FIRST-ORDER METHODS

Feature	Prox. point	FGM	AMGS	FISTA	FISTA-CP	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	almost	yes	yes

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 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 that of FISTA-CP, for strongly convex f and non-strongly convex Ψ, that of scAPG and, for $\Psi = 0$, that of FGM. 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.

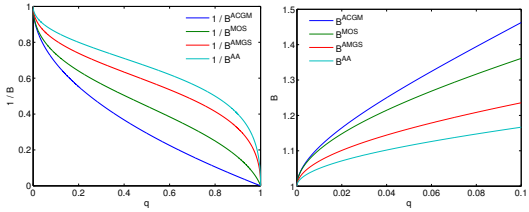
For ACGM, MOS, and AA, we have

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

where

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

[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

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 Figure 3. The inverse rates are compared for every possible value of q in Figure 3(a) whereas the rates are compared directly in Figure 3(b) for the range of q found in the vast majority of practical applications.

It can be clearly discerned from (44), (45), and Figure 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 $Q_{f',L+\mu_\Psi,\boldsymbol{y}}$ using the definition of Q in (1) and the strong convexity transfer in (12) we obtain

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

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 $\boldsymbol{x} \in \mathbb{R}^n$. 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

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

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 $\boldsymbol{x} \in \mathbb{R}^n$. 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

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

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

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

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

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

with

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

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

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

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

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

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:

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

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

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

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

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

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

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

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

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

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

Appendix E
Proof of Theorem 2

In the non-strongly convex case, we have

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

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

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

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

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

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

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

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

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

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

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

Using $A_1 = \frac{1}{L_1-\mu_f} \ge \frac{1}{L_u-\mu_f}$, the strongly convex 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

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

Replacing (42) in ACGM with

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

results in an algorithm that produces identical iterates to FISTA. The convergence analysis of ACGM employing (70) instead of (42) yields 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*. 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, Sept. 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, Sept. 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] ——, "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] ——, *Introductory Lectures on Convex Optimization. Applied Optimization, vol. 87*. Boston, MA: Kluwer Academic Publishers, 2004.

[7] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imaging 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. (2017, May) Estimate sequence methods: extensions and approximations. [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. Imaging 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 Software*, 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, 2009.

[14] A. Nemirovski and D.-B. Yudin, *Problem complexity and method efficiency in optimization*. New York, NY: John Wiley & Sons, 1983.

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

[16] N. Parikh, S. P. Boyd *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, 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 *ICML*, 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: 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. R. Stat. Soc. Ser. 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. of IEEE Intern. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, March 2017, New Orleans, USA, pp. 4521–4525.

[28] P. C. Hansen, J. G. Nagy, and D. P. O'Leary, *Deblurring images: matrices, spectra, and filtering*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2006.

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

[30] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *J. R. Stat. Soc. Ser. 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.

Publication III

Mihai I. Florea and Sergiy A. Vorobyov. A Generalized Accelerated Composite Gradient Method: Uniting Nesterov's Fast Gradient Method and FISTA. Submitted to *IEEE Transactions on Signal Processing*, Oct. 2018.

A Generalized Accelerated Composite Gradient Method: Uniting Nesterov's Fast Gradient Method and FISTA

Mihai I. Florea and Sergiy A. Vorobyov

***Abstract*—Numerous problems in signal processing, statistical inference, computer vision, and machine learning, can be cast as large-scale convex optimization problems. Due to their size, many of these problems can only be addressed by first-order black-box methods. The most popular among these are the Fast Gradient Method (FGM) and the Fast Iterative Shrinkage Thresholding Algorithm (FISTA). FGM requires that the objective be finite differentiable with known Lipschitz constant. FISTA is applicable to the more broad class of composite objectives and is equipped with a line-search procedure for estimating the Lipschitz constant. Nonetheless, FISTA cannot increase the step size and is unable to take advantage of strong convexity. FGM and FISTA are very similar in form. Despite this, they appear to have vastly differing convergence analyses. In this work we generalize the previously introduced augmented estimate sequence framework as well as the related notion of the gap sequence. We showcase the flexibility of our tools by constructing a Generalized Accelerated Composite Gradient Method, that unites FGM and FISTA, along with their most popular variants. We further showcase the flexibility of our tools by endowing our method with monotonicity alongside a versatile line-search procedure. By simultaneously incorporating the strengths of FGM and FISTA, our method is able to surpass both in terms of robustness and usability. From a theoretical perspective, the Lyapunov property of the generalized gap sequence used in deriving our method implies that both FGM and FISTA are amenable to a Lyapunov analysis, common among optimization algorithms. We support our findings with simulation results on an extensive benchmark of composite problems. Our experiments show that monotonicity has a stabilizing effect on convergence and challenge the notion present in the literature that for strongly convex objectives, accelerated proximal schemes can be reduced to fixed momentum methods.**

***Index Terms*—estimate sequence, Nesterov method, fast gradient method, FISTA, monotone, line-search, composite objective, large-scale optimization**

I. Introduction

Numerous large-scale convex optimization problems have recently emerged in a variety of fields, including signal and image processing, statistical inference, computer vision, and machine learning. Often, little is known about the actual structure of the objective function. Therefore, optimization algorithms used in solving such problems can only rely (e.g., by means of callback functions) on specific black-box methods, called oracle functions [1]. The term "large-scale" refers to the tractability of certain computational primitives (see also [2]). In the black-box setting, it means that the oracle functions of large-scale problems usually only include scalar functions and operations that resemble first-order derivatives.

Many large-scale applications were rendered practical to address with the advent of Nesterov's Fast Gradient Method (FGM) [3]. FGM requires that the objective be differentiable with Lipschitz gradient. Many optimization problems, particularly inverse problems in fields such as sparse signal processing, linear algebra, matrix and tensor completion, and digital imaging (see [4]–[8] and references therein), have a composite structure. In these composite problems, the objective F is the sum of a function f with Lipschitz gradient (Lipschitz constant L_f) and a simple but possibly non-differentiable regularizer Ψ. The regularizer Ψ embeds constraints by being infinite outside the feasible set. Often, L_f is not known in advance. The composite problem oracle functions are the scalar $f(\boldsymbol{x})$ and $\Psi(\boldsymbol{x})$, as well as the gradient $\nabla f(\boldsymbol{x})$ and the proximal operator $prox_{\tau\Psi}(\boldsymbol{x})$.

To address composite problems, Nesterov has devised an Accelerated Multistep Gradient Scheme (AMGS) [9]. This method updates a Lipschitz constant estimate (LCE) at every iteration using a subprocess commonly referred in the literature as "line-search" [7], [10]. The generation of a new iterate (advancement phase of an iteration) and line-search are interdependent and cannot be executed in parallel. Moreover, AMGS utilizes only the gradient-type oracle functions $\nabla f(\boldsymbol{x})$ and $prox_{\tau\Psi}(\boldsymbol{x})$. In many applications, including compressed sensing (e.g., LASSO [11]) and many classification tasks (e.g., l_1-regularized logistic regression), the evaluation of $\nabla f(\boldsymbol{x})$ is more computationally expensive than $f(\boldsymbol{x})$. An alternative to AMGS that uses $f(\boldsymbol{x})$ calls in line-search has been proposed by Beck and Teboulle in the form of the Fast Iterative Shrinkage-Thresholding Algorithm (FISTA) [10]. FISTA also benefits from having line-search decoupled from advancement. However, FISTA is unable to decrease the LCE at run-time. A strongly convex extension of FISTA, which we designate as FISTA Chambolle-Pock (FISTA-CP), has been recently introduced in [6], but without line-search. An unrelated study [12] proposes a variant of FISTA-CP equipped with a line-search procedure capable of decreasing the LCE (fully adaptive line-search). However, this strongly convex Accelerated Proximal Gradient (scAPG) method is guaranteed to converge only if f is strongly convex.

FGM was derived using the estimate sequence [13]. This flexible framework was adapted in [9] to include AMGS as well. FISTA-CP (and FISTA when the objective is non-strongly convex) is identical in form to FGM. Therefore,

M. I. Florea (E-mail: mihai.florea@aalto.fi) and S. A. Vorobyov (E-mail: sergiy.vorobyov@aalto.fi) are with Aalto University, Dept. Signal Processing and Acoustics, FI-00076, AALTO, Finland. This work has been partially supported by the Academy of Finland grant No. 299243.

FISTA and FISTA-CP can be viewed as an extension of FGM for composite objectives. However, the convergence analyses of FISTA and FISTA-CP, each different from the other, do not appear to involve the estimate sequence at all. Consequently, new features of FGM cannot be directly incorporated into FISTA and FISTA-CP. Recently, Nesterov has proposed in [14] a line-search variant of FGM, albeit only for non-strongly convex objectives. Neither a derivation nor a convergence analysis have been provided, but can be readily obtained using the estimate sequence framework. Had FISTA utilized the same convergence analysis as FGM, a fully adaptive line-search variant of FISTA could simply take the form in [14]. Instead, a sophisticated fully adaptive line-search extension was proposed in [15], with a technical derivation based on the mathematical constructs of [10]. However, through partial adoption of the estimate sequence, i.e., relating FISTA to "constant step scheme I" in [13] and AMGS, we have arrived at a similar but simpler fully adaptive line-search scheme for FISTA [16], but again only in the non-strongly convex scenario.

In [17], we have introduced the *augmented estimate sequence* framework and used it to derive the Accelerated Conjugate Gradient Method (ACGM), which incorporates by design a fully adaptive line-search procedure. ACGM has the convergence guarantees of FGM, the best among primal first-order methods, while being as broadly applicable as AMGS. In addition, FISTA-CP and FISTA, along with the fully adaptive line-search extensions in [15] and [16], are particular cases of ACGM [17]. However, to accommodate infeasible start, we have imposed restrictions on the input parameters. Variants of FGM (e.g., "constant step scheme III" in [13]) exist that are guaranteed to converge only when the starting point is feasible, and thus do not correspond to any instance of ACGM.

In this paper, we generalize the augmented estimate sequence framework and derive a generalization of ACGM that encompasses FGM, FISTA, and FISTA-CP, along with their variants. We further showcase the flexibility and power of the augmented estimate sequence framework by endowing ACGM with monotonicity alongside its adaptive line-search procedure. Monotonicity is a desirable property, particularly when dealing with proximal operators that lack a closed form expression or other kinds of inexact oracles [6], [18]. Even when dealing with exact oracles, monotonicity leads to a more stable and predictable convergence rate. The resulting generalized ACGM is therefore superior to FGM and FISTA in terms of flexibility and usability. We support our theoretical findings with simulation examples.

A. Assumptions and notation

Consider composite optimization problems of the form

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} F(\boldsymbol{x}) \stackrel{\text{def}}{=} f(\boldsymbol{x}) + \Psi(\boldsymbol{x}),$$

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

Apart from the above properties, nothing is assumed known about functions f and Ψ, which can only be accessed in a *black-box* fashion [1] by querying 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 $\tau > 0$. The proximal operator $\text{prox}_{\tau\Psi}(\boldsymbol{x})$ is given by

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

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

Central to our derivation are *generalized parabolae*, quadratic functions whose Hessians are multiples of the identify matrix. We refer to the strongly convex ones simply as *parabolae*, of the form $\psi : \mathbb{R}^n \to \mathbb{R}$,

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

where $\gamma > 0$ denotes the curvature, $\boldsymbol{v}$ is the vertex, and ψ^* is the optimum value.

For conciseness, we introduce the generalized parabola expression $Q_{f,\gamma,\boldsymbol{y}}(\boldsymbol{x})$ for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ and $\gamma \geq 0$ as

$$Q_{f,\gamma,\boldsymbol{y}}(\boldsymbol{x}) \stackrel{\text{def}}{=} f(\boldsymbol{y}) + \langle \nabla f(\boldsymbol{y}), \boldsymbol{x}-\boldsymbol{y}\rangle + \frac{\gamma}{2}\|\boldsymbol{x}-\boldsymbol{y}\|_2^2. \quad (1)$$

The proximal gradient operator $T_L(\boldsymbol{y})$ [9] can be expressed succinctly using (1) as

$$T_L(\boldsymbol{y}) \stackrel{\text{def}}{=} \operatorname*{arg\,min}_{\boldsymbol{x}\in\mathbb{R}^n} \left(Q_{f,L,\boldsymbol{y}}(\boldsymbol{x}) + \Psi(\boldsymbol{x})\right), \quad \boldsymbol{y} \in \mathbb{R}^n, \quad (2)$$

where $L > 0$ is a parameter corresponding to the inverse of the step size. Within the scope of this work, the left-hand side of (2) does not need functional parameters. Operator $T_L(\boldsymbol{y})$ can be evaluated in terms of oracle functions as

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

The composite gradient [9] is given by

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

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

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

II. Generalizing ACGM

A. Nesterov's first order method design pattern

Nesterov's FGM and AMGS adhere to the design pattern outlined in Algorithm 1 (early variant discussed in [17]). This pattern will form the scaffolding of our generalized ACGM.

Algorithm 1 takes as input the starting point $\boldsymbol{x}_0$, function ψ_0 and, if the Lipschitz constant is not known in advance, an

initial LCE $L_0 > 0$. As we shall see later on, ψ_0 is the initial estimate function within the generalized augmented sequence framework (Subsection II-C). The pattern in Algorithm 1 fits within the family of majorization-minimization algorithms. In line 5 of Algorithm 1, the main iterate is given by the minimum of $u_{k+1}(\boldsymbol{x})$, a *local* upper bound (at $\boldsymbol{x}_{k+1}$) on the objective F. This upper bound is uniquely determined by an auxiliary point $\boldsymbol{y}_{k+1}$. Alongside the main iterate, the algorithm maintains an estimate function ψ_{k+1}, obtained from the previous one by adding a *global* lower bound $w_{k+1}(\boldsymbol{x})$ weighted by $a_{k+1} > 0$ (line 6 of Algorithm 1). The current LCE L_{k+1}, weight a_{k+1}, and auxiliary point $\boldsymbol{y}_{k+1}$ are computed using algorithm specific methods $\mathcal{S}$, $\mathcal{F}_a$, and $\mathcal{F}_y$, respectively (lines 2, 3, and 4 of Algorithm 1). These methods take as parameters the state of the algorithm, given by current values of the main iterate, LCE, weight, and estimate function.

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

1: **for** $k = 0, \ldots, K-1$ **do**
2: $\quad L_{k+1} = \mathcal{S}(\boldsymbol{x}_k, \psi_k, L_k)$
3: $\quad a_{k+1} = \mathcal{F}_a(\psi_k, L_{k+1})$
4: $\quad \boldsymbol{y}_{k+1} = \mathcal{F}_y(\boldsymbol{x}_k, \psi_k, a_{k+1})$
5: $\quad \boldsymbol{x}_{k+1} = \arg\min_{\boldsymbol{x}\in\mathbb{R}^n} u_{k+1}(\boldsymbol{x})$
6: $\quad \psi_{k+1}(\boldsymbol{x}) = \psi_k(\boldsymbol{x}) + a_{k+1} w_{k+1}(\boldsymbol{x})$
7: **end for**

B. FGM Estimate Sequence

When the objective function is strongly convex, many first-order schemes, including the non-accelerated fixed-point methods, guarantee linear convergence of iterates to the optimal point. When the problem is non-strongly convex, the optimization landscape may contain a high-dimensional subspace of very low curvature in the vicinity of the set of optimal points. In this case, the convergence of iterates remains a difficult open problem [19]. For instance, Nesterov has provided in [13] an ill-conditioned quadratic problem where the convergence of iterates to an optimal point is intractable for all first-order schemes of a certain structure.

Hence, we choose to measure convergence using the image space distance (ISD), which is the distance between the objective values at iterates and the optimal value. The decrease rate of an upper bound on the ISD gives the convergence guarantee (provable convergence rate). The estimate sequence framework follows naturally from the formulation of such guarantees. Specifically, we interpret the image space distance upper bound (ISDUB), provided by Nesterov for FGM in [13], for all $k \geq 0$ as

$$A_k(F(\boldsymbol{x}_k) - F(\boldsymbol{x}^*)) \leq A_0(F(\boldsymbol{x}_0) - F(\boldsymbol{x}^*)) + \frac{\gamma_0}{2}\|\boldsymbol{x}_0 - \boldsymbol{x}^*\|_2^2. \tag{3}$$

Point $\boldsymbol{x}^*$ can be any optimal point. However, we consider it fixed throughout this work. The convergence guarantee is given by the sequence $\{A_k\}_{k\geq 0}$ with $A_0 \geq 0$ and $A_k > 0$ for all $k \geq 1$. The right-hand side of (3) is a weighted sum between the initial ISD and the corresponding domain space term (DST), with weights given by A_0 and γ_0, respectively. In the derivation of FGM, the weights are constrained as $A_0 > 0$ and $\gamma_0 \geq A_0\mu$. When the starting point $\boldsymbol{x}_0$ is not guaranteed to be feasible, A_0 must be zero. For AMGS, γ_0 is fixed as 1 while for original ACGM [17], to prevent A_0 from being unbounded above, we have enforced $\gamma_0 \leq 1$. Given that our current aim is to provide a generic framework, we impose no restrictions on the weights, apart from $A_0 \geq 0$ and $\gamma_0 > 0$. The former restriction follows from the convexity of F while the latter is required by the estimate sequence, along with its augmented variant, as we shall demonstrate in the sequel.

The ISDUB expression can be rearranged to take the form

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

where

$$H_k \stackrel{\text{def}}{=} (A_k - A_0)F(\boldsymbol{x}^*) + A_0 F(\boldsymbol{x}_0) + \frac{\gamma_0}{2}\|\boldsymbol{x}_0 - \boldsymbol{x}^*\|_2^2$$

is the highest upper bound that can be placed on weighted objective values $A_k F(\boldsymbol{x}_k)$ to satisfy (3).

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

$$F(\boldsymbol{x}) \geq W_k(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^n, \quad k \geq 1. \tag{5}$$

By substituting the optimal value terms $F(\boldsymbol{x}^*)$ in (4) with $W_k(\boldsymbol{x}^*)$, we obtain $\mathcal{H}_k$, a lower bound on H_k, given by

$$\mathcal{H}_k \stackrel{\text{def}}{=} (A_k - A_0)W_k(\boldsymbol{x}^*) + A_0 F(\boldsymbol{x}_0) + \frac{\gamma_0}{2}\|\boldsymbol{x}^* - \boldsymbol{x}_0\|_2^2,$$

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

$$\psi_k(\boldsymbol{x}) \stackrel{\text{def}}{=} (A_k - A_0)W_k(\boldsymbol{x}) + A_0 F(\boldsymbol{x}_0) + \frac{\gamma_0}{2}\|\boldsymbol{x} - \boldsymbol{x}_0\|_2^2, \tag{6}$$

for all $\boldsymbol{x} \in \mathbb{R}^n$ and $k \geq 0$. Note that the first estimate function ψ_0 does not contain a lower bound term. Therefore, it is not necessary to define W_0. The collection of estimate functions $\{\psi_k(\boldsymbol{x})\}_{k\geq 0}$, is referred to as the *estimate sequence*.

The estimate function optimum value, given by

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

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

$$\psi_k^* = \min_{\boldsymbol{x}\in\mathbb{R}^n} \psi_k(\boldsymbol{x}) \leq \psi_k(\boldsymbol{x}^*) = \mathcal{H}_k, \quad k \geq 0. \tag{7}$$

As such, ψ_k^* provides the sought after computable replacement of H_k. Note that if the lower bounds W_k are linear, the estimate functions are generalized parabolae with the curvature given by γ_0. In this case, the existence of ψ_k^* is conditioned by $\gamma_0 > 0$, explaining the assumption made in (3).

Thus, it suffices to maintain the augmented estimate sequence property, given by

$$A_k F(\boldsymbol{x}_k) \leq \psi_k^*, \quad k \geq 0, \tag{8}$$

to satisfy the ISDUB expression (3). The proof follows from the above definitions as

$$A_k F(\boldsymbol{x}_k) \overset{(8)}{\leq} \psi_k^* \overset{(7)}{\leq} \psi_k(\boldsymbol{x}^*) = \mathcal{H}_k \overset{(5)}{\leq} H_k, \quad k \geq 0.$$

C. Generalizing the Augmented Estimate Sequence

The interval between the maintained upper bound ψ_k^* and the highest allowable bound H_k contains $\mathcal{H}_k$. This allows us to produce a relaxation of the estimate sequence by forcibly *closing the gap* between $\mathcal{H}_k$ and H_k. Namely, we define the augmented estimate functions as

$$\psi'_k(\boldsymbol{x}) \overset{\text{def}}{=} \psi_k(\boldsymbol{x}) + H_k - \mathcal{H}_k, \quad k \geq 0, \tag{9}$$

with $\{\psi'_k(\boldsymbol{x})\}_{k\geq 0}$ being the augmented estimate sequence. We expand definition (9) as

$$\psi'_k(\boldsymbol{x}) = \psi_k(\boldsymbol{x}) + (A_k - A_0)(F(\boldsymbol{x}^*) - W_k(\boldsymbol{x}^*)).$$

The augmented estimate sequence property is thus given by

$$A_k F(\boldsymbol{x}_k) \leq \psi'^*_k.$$

D. Lower bounds

As mentioned in Subsection II-A, we can construct an optimization scheme based on the design pattern in Algorithm 1. The pattern requires us to specify at every iteration a global lower bound $w_{k+1}(\boldsymbol{x})$, a local upper bound $u_{k+1}(\boldsymbol{x})$, update rules $\mathcal{F}_a$, and $\mathcal{F}_y$, and line-search $\mathcal{S}$.

We set the lower bounds to be supporting generalized parabolae, namely

$$w_{k+1}(\boldsymbol{x}) = \mathcal{R}_{L_{k+1}+\mu_\Psi, \boldsymbol{y}_{k+1}}(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^n, \quad k \geq 0. \tag{10}$$

Lemma 1. *Supporting generalized parabolae* (10) *are valid global lower bounds on the objective F if the auxiliary points and LCEs obey the descent condition, stated as*

$$f(\boldsymbol{z}_{k+1}) \leq Q_{f, L_{k+1}, \boldsymbol{y}_{k+1}}(\boldsymbol{z}_{k+1}), \quad k \geq 0, \tag{11}$$

where

$$\boldsymbol{z}_{k+1} \overset{\text{def}}{=} T_{L_{k+1}}(\boldsymbol{y}_{k+1}). \tag{12}$$

Proof: Same as [17, Lemma 2], with $\boldsymbol{x}_{k+1}$ replaced by $\boldsymbol{z}_{k+1}$. ∎

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

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

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

$$a_{k+1} > 0, \tag{13}$$

$$A_{k+1} = A_k + a_{k+1}, \tag{14}$$

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

From definition (6), we have that the initial estimate function is a parabola, given by

$$\psi_0(\boldsymbol{x}) = A_0 F(\boldsymbol{x}_0) + \frac{\gamma_0}{2}\|\boldsymbol{x} - \boldsymbol{x}_0\|_2^2. \tag{15}$$

From (10) and (13), line 6 of Algorithm 1 further ensures that estimate functions at every iteration are parabolic. We write the estimate functions and the augmented estimate functions, for all $k \geq 0$, as

$$\psi_k(\boldsymbol{x}) = \psi_k^* + \frac{\gamma_k}{2}\|\boldsymbol{x} - \boldsymbol{v_k}\|_2^2, \tag{16}$$

$$\psi'_k(\boldsymbol{x}) = \psi'^*_k + \frac{\gamma_k}{2}\|\boldsymbol{x} - \boldsymbol{v_k}\|_2^2, \tag{17}$$

with

$$\psi'^*_k = \psi_k^* + (A_k - A_0)(F(\boldsymbol{x}^*) - W_k(\boldsymbol{x}^*)). \tag{18}$$

The estimate sequence update in line 6 of Algorithm 1 along with (10), (12), and (16) gives, through differentiation and coefficient matching (see also [17]), update rules for the estimate sequence curvatures and vertices, for all $k \geq 0$, as

$$\gamma_{k+1} = \gamma_k + a_{k+1}\mu, \tag{19}$$

$$\begin{aligned}\boldsymbol{v}_{k+1} = \frac{1}{\gamma_{k+1}}&(\gamma_k \boldsymbol{v}_k + a_{k+1}(L_{k+1} + \mu_\Psi)\boldsymbol{z}_{k+1} \\ &- a_{k+1}(L_{k+1} - \mu_f)\boldsymbol{y}_{k+1}).\end{aligned} \tag{20}$$

It follows from (19) that the estimate function curvature can obtained directly from the cumulative weights as

$$\gamma_k = \gamma_0 + \left(\sum_{i=1}^{k} a_i\right)\mu = \gamma_0 - A_0\mu + A_k\mu, \quad k \geq 0. \tag{21}$$

E. Generalizing the Gap Sequence

A sufficient condition for the preservation of the augmented estimate sequence property (8) as the algorithm progresses is that the augmented estimate sequence gap, defined as

$$\Gamma_k \overset{\text{def}}{=} A_k F(\boldsymbol{x}_k) - \psi'^*_k, \quad k \geq 0, \tag{22}$$

is non-increasing. When the estimate functions are parabolic, this gap can be written as

$$\begin{aligned}\Gamma_k &\overset{(18)}{=} A_k(F(\boldsymbol{x}_k) - F(\boldsymbol{x}^*)) + (A_k - A_0)W_k(\boldsymbol{x}^*) \\ &\quad + A_0 F(\boldsymbol{x}^*) - \psi_k^* \\ &\overset{(16)}{=} A_k(F(\boldsymbol{x}_k) - F(\boldsymbol{x}^*)) + \frac{\gamma_k}{2}\|\boldsymbol{x}^* - \boldsymbol{v_k}\|_2^2 - \\ &\quad - \psi_k(\boldsymbol{x}^*) + A_0 F(\boldsymbol{x}^*) + (A_k - A_0)W_k(\boldsymbol{x}^*) \\ &\overset{(6)}{=} A_k(F(\boldsymbol{x}_k) - F(\boldsymbol{x}^*)) + \frac{\gamma_k}{2}\|\boldsymbol{v_k} - \boldsymbol{x}^*\|_2^2 - \\ &\quad - A_0(F(\boldsymbol{x}_0) - F(\boldsymbol{x}^*)) - \frac{\gamma_0}{2}\|\boldsymbol{x_0} - \boldsymbol{x}^*\|_2^2, \quad k \geq 0.\end{aligned}$$

We introduce the gap sequence $\{\Delta_k\}_{k\geq 0}$ in the form

$$\Delta_k \overset{\text{def}}{=} A_k(F(\boldsymbol{x}_k) - F(\boldsymbol{x}^*)) + \frac{\gamma_k}{2}\|\boldsymbol{v}_k - \boldsymbol{x}^*\|_2^2, \quad k \geq 0. \tag{23}$$

The augmented estimate sequence gaps can be expressed more succinctly as

$$\Gamma_k = \Delta_k - \Delta_0, \quad k \geq 0. \tag{24}$$

Hence, the variation of the two sequences is identical, with the only difference being that the augmented estimate sequence

gap is constrained to be zero initially. The sufficient condition becomes

$$\Delta_{k+1} \leq \Delta_k, \quad k \geq 0. \tag{25}$$

F. Upper bounds

Interestingly, all of the above results do not rely on a specific form of the local upper bound $u_{k+1}(\boldsymbol{x})$, as long as assumption (11) holds for all $k \geq 0$. We want our algorithm to converge as fast as possible while maintaining the monotonicity property, expressed as

$$F(\boldsymbol{x}_{k+1}) \leq F(\boldsymbol{x}_k), \quad k \geq 0. \tag{26}$$

Then, without further knowledge of the objective function, (12) and (26) suggest a simple expression of the upper bound for all $k \geq 0$ in the form of

$$u_{k+1}(\boldsymbol{x}) = \min\{Q_{f,L_{k+1},\boldsymbol{y}_{k+1}}(\boldsymbol{x}) + \Psi(\boldsymbol{x}), F(\boldsymbol{x}_k) + \sigma_{\{\boldsymbol{x}_k\}}(\boldsymbol{x})\}, \tag{27}$$

where σ_X is the indicator function [20] of set X, given by

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

G. Towards an algorithm

With all building blocks in place, we select functions $\mathcal{S}$, $\mathcal{F}_a$, and $\mathcal{F}_y$ so as to preserve the Lyapunov property of the gap sequence (25).

The enforced descent condition (11) can be *equivalently* expressed in terms of composite objective values as

$$F(\boldsymbol{z}_{k+1}) \leq Q_{f,L_{k+1},\boldsymbol{y}_{k+1}}(\boldsymbol{z}_{k+1}) + \Psi(\boldsymbol{z}_{k+1}), \quad \boldsymbol{x} \in \mathbb{R}^n.$$

In [17, Theorem 3] we have shown that if we choose $\boldsymbol{x}_{k+1} = \boldsymbol{z}_{k+1}$, we can build a method that maintains a monotone gap sequence. In this work we need a stronger result that allows $\boldsymbol{x}_{k+1}$ to be "better" than $\boldsymbol{z}_{k+1}$.

Theorem 1. *If at iteration $k \geq 0$ we have*

$$F(\boldsymbol{x}_{k+1}) \leq F(\boldsymbol{z}_{k+1}) \leq Q_{f,L_{k+1},\boldsymbol{y}_{k+1}}(\boldsymbol{z}_{k+1}) + \Psi(\boldsymbol{z}_{k+1}),$$

then

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

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

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

Proof: See Appendix A. ∎

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

$$\boldsymbol{y}_{k+1} = \mathcal{F}_y(\boldsymbol{x}_k, \psi_k, A_k, a_{k+1}) = \frac{A_k\gamma_{k+1}\boldsymbol{x}_k + a_{k+1}\gamma_k\boldsymbol{v}_k}{A_k\gamma_{k+1} + a_{k+1}\gamma_k}, \tag{28}$$

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

The latter, combined with the non-negativity of the weights (13), yields

$$a_{k+1} \leq \mathcal{E}(\gamma_k, A_k, L_{k+1}), \quad k \geq 0, \tag{29}$$

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

$$\mathcal{E}(\gamma_k, A_k, L_{k+1}) \stackrel{\text{def}}{=} \frac{1}{2(L_{k+1}-\mu_f)} \left(\gamma_k + A_k\mu + \sqrt{(\gamma_k + A_k\mu)^2 + 4(L_{k+1}-\mu_f)A_k\gamma_k}\right).$$

The cumulative weight A_k in (14) gives the convergence guarantee in (3). To provide the best guarantees, we enforce equality in (29), namely

$$a_{k+1} = \mathcal{F}_a(\psi_k, A_k, L_{k+1}) = \mathcal{E}(\gamma_k, A_k, L_{k+1}). \tag{30}$$

For determining the LCE, we select the backtracking line-search method $\mathcal{S}_A$ employed by AMGS [9] and the original ACGM [17]. The search parameters comprise the LCE increase rate $r_u > 1$ and the LCE decrease rate $0 < r_d < 1$. The search terminates when the line-search stopping criterion (LSSC) in (11) is satisfied.

H. Putting it all together

We have thus determined a search strategy $\mathcal{S}_A$, initial estimate function ψ_0 in (15), upper bounds $u_{k+1}(\boldsymbol{x})$ in (27), lower bounds $w_{k+1}(\boldsymbol{x})$ in (10), function $\mathcal{F}_a$ in (30), and function $\mathcal{F}_y$ in (28). Substituting these expressions in the design pattern outlined in Algorithm 1, we can write down a generalization of ACGM in estimate sequence form, as listed in Algorithm 2.

Non-monotone generalized ACGM can be obtained by enforcing $\boldsymbol{x}_{k+1} = \boldsymbol{z}_{k+1}$ for all $k \geq 0$, accomplished by replacing line 16 of Algorithm 2 with

$$\boldsymbol{x}_{k+1} := \hat{\boldsymbol{z}}_{k+1}. \tag{31}$$

III. Complexity analysis

A. Worst-case convergence guarantees

Algorithm 2 maintains the convergence guarantee in (3) explicitly at run-time as state variable A_k. Moreover, if sufficient knowledge of the problem is available, it is possible to formulate a worst-case convergence guarantee *before* running the algorithm.

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

$$\begin{aligned}
q_{k+1} &\stackrel{\text{def}}{=} \frac{\mu}{L_{k+1}+\mu_\Psi}, \\
L_u &\stackrel{\text{def}}{=} \max\{r_u L_f, r_d L_0\}, \\
q_u &\stackrel{\text{def}}{=} \frac{\mu}{L_u+\mu_\Psi}.
\end{aligned}$$

The worst-case convergence guarantees for generalized ACGM are stated in following theorem.

Algorithm 2 Generalized monotone ACGM in estimate sequence form

ACGM($\boldsymbol{x_0}$, L_0, μ_f, μ_Ψ, A_0, γ_0, r_u, r_d, K)

1: $\boldsymbol{v}_0 = \boldsymbol{x}_0,\ \mu = \mu_f + \mu_\Psi$
2: **for** $k = 0, \ldots, K-1$ **do**
3: $\quad \hat{L}_{k+1} := r_d L_k$
4: $\quad$ **loop**
5: $\quad\quad \hat{a}_{k+1} := \frac{1}{2(\hat{L}_{k+1}-\mu_f)} \left(\gamma_k + A_k\mu + \sqrt{(\gamma_k + A_k\mu)^2 + 4(\hat{L}_{k+1}-\mu_f)A_k\gamma_k}\right)$
6: $\quad\quad \hat{A}_{k+1} := A_k + \hat{a}_{k+1}$
7: $\quad\quad \hat{\gamma}_{k+1} := \gamma_k + \hat{a}_{k+1}\mu$
8: $\quad\quad \hat{\boldsymbol{y}}_{k+1} := \frac{1}{A_k\hat{\gamma}_{k+1} + \hat{a}_{k+1}\gamma_k}(A_k\hat{\gamma}_{k+1}\boldsymbol{x}_k + \hat{a}_{k+1}\gamma_k\boldsymbol{v}_k)$
9: $\quad\quad \hat{\boldsymbol{z}}_{k+1} := \mathrm{prox}_{\frac{1}{\hat{L}_{k+1}}\Psi}\left(\hat{\boldsymbol{y}}_{k+1} - \frac{1}{\hat{L}_{k+1}}\nabla f(\hat{\boldsymbol{y}}_{k+1})\right)$
10: $\quad\quad$ **if** $f(\hat{\boldsymbol{z}}_{k+1}) \le Q_{f,\hat{L}_{k+1},\hat{\boldsymbol{y}}_{k+1}}(\hat{\boldsymbol{z}}_{k+1})$ **then**
11: $\quad\quad\quad$ Break from loop
12: $\quad\quad$ **else**
13: $\quad\quad\quad \hat{L}_{k+1} := r_u\hat{L}_{k+1}$
14: $\quad\quad$ **end if**
15: $\quad$ **end loop**
16: $\quad \boldsymbol{x}_{k+1} := \arg\min\{F(\hat{\boldsymbol{z}}_{k+1}), F(\boldsymbol{x}_k)\}$
17: $\quad \boldsymbol{v}_{k+1} := \frac{1}{\hat{\gamma}_{k+1}}(\gamma_k\boldsymbol{v}_k + \hat{a}_{k+1}(\hat{L}_{k+1}+\mu_\Psi)\hat{\boldsymbol{z}}_{k+1} - \hat{a}_{k+1}(\hat{L}_{k+1}-\mu_f)\hat{\boldsymbol{y}}_{k+1})$
18: $\quad L_{k+1} := \hat{L}_{k+1},\ A_{k+1} := \hat{A}_{k+1},\ \gamma_{k+1} := \hat{\gamma}_{k+1}$
19: **end for**
20: **return** x_K

Theorem 2. *If $\gamma_0 \ge A_0\mu$, the generalized ACGM algorithm generates a sequence $\{\boldsymbol{x}_k\}_{k\ge1}$ that satisfies*

$$F(\boldsymbol{x}_k) - F(\boldsymbol{x}^*) \le \min\left\{\frac{4}{(k+1)^2}, (1-\sqrt{q_u})^{k-1}\right\}(L_u - \mu_f)\bar{\Delta}_0, \quad k \ge 1,$$

where

$$\bar{\Delta}_0 \stackrel{\text{def}}{=} \frac{\Delta_0}{\gamma_0} = \frac{A_0}{\gamma_0}(F(\boldsymbol{x}_0) - F(\boldsymbol{x}^*)) + \frac{1}{2}\|\boldsymbol{x}_0 - \boldsymbol{x}^*\|_2^2.$$

Proof: See Appendix B. ∎

Note that the assumption $\gamma_0 \ge A_0\mu$ always holds for non-strongly convex objectives and that ACGM is guaranteed converge for strongly convex objectives also when $\gamma_0 < A_0\mu$. However, in the latter case, it is more difficult to obtain simple lower bounds on the convergence guarantees. We leave such an endeavor to future research.

B. Wall-clock time units

So far, we have measured the theoretical performance of algorithms in terms of convergence guarantees (including the worst-case ones) indexed in iterations. This does not account for the complexity of individual iterations. In [17], we have introduced a new measure of complexity, the *wall-clock time unit* (WTU), to compare optimization algorithms more reliably. We thus distinguish between two types of convergence guarantees. One is the previously used *iteration convergence guarantee*, indexed in iterations and a new *computational convergence guarantee*, indexed in WTU.

The WTU is a measure of running time in a *shared memory parallel scenario*. The computing environment consists of a small number of parallel processing units (PPU). Each PPU may be a parallel machine itself. The number of parallel units is considered sufficient to compute any number of independent oracle functions simultaneously. The shared-memory system does not impose constraints on parallelization, namely, it is uniform memory access (UMA) [21] and it is large enough to store the arguments and results of oracle calls for as long as they are needed.

In order to compare algorithms based on a unified benchmark, in [17] we have assumed that f and ∇f require 1 WTU each while all other operations are negligible and amount to 0 WTU. In this work, we generalize the analysis. We attribute finite non-negative costs t_f, t_g, t_Ψ, and t_p to $f(\boldsymbol{x})$, $\nabla f(\boldsymbol{x})$, $\Psi(\boldsymbol{x})$, and $\mathrm{prox}_{r\Psi}(\boldsymbol{x})$, respectively. However, since we are dealing with large-scale problems, we maintain the assumption that element-wise vector operations, including scalar-vector multiplications, vector additions, and inner products, have negligible complexity when compared to oracle functions and assign a cost of 0 WTU to each. Synchronization of PPUs also incurs no cost. Consequently, when computed in isolation, an objective function value $F(\boldsymbol{x})$ call costs $t_F = \max\{t_f, t_\Psi\}$, ascribable to separability, while a proximal gradient operation costs $t_T = t_g + t_p$, due to computational dependencies.

C. Per-iteration complexity

We measure this complexity in WTU on the shared memory system described in the previous subsection and consider a parallel implementation involving speculative execution [21].

The advancement phase of a generalized ACGM iteration consists of one proximal gradient step (line 9 of Algorithm 2). Hence, every iteration has a base cost of $t_T = t_g + t_p$. The LSSC and the monotonicity condition (MC) in line 16 of Algorithm 2 can be evaluated in parallel with subsequent iterations. Both rely on the computation of $f(\hat{\boldsymbol{z}}_{k+1})$, which in the worst case requires $\lceil t_f/t_T \rceil$ dedicated PPUs. In addition, MC may need up to $\lceil t_\Psi/t_T \rceil$ PPUs.

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

Overshoots are assumed to occur even less often. Similarly, the algorithm proceeds speculating that MC always passes

and defaults to (31). Hence, MC has $t_d = t_F$, due to its dependency on $\Psi(\hat{\boldsymbol{z}}_{k+1})$, but once the algorithmic state of iteration k has been restored, no additional oracle calls are needed, leading to $t_c = 0$. MC and LSSC can be fused into a single condition, giving rise to the scenarios outlined in Table I. Note that if LSSC fails, MC is not evaluated.

TABLE I
ALGORITHM STALL TIME IN WTU BASED ON THE OUTCOME OF LSSC AND MC

	MC passed	MC failed
LSSC passed	0	$\max\{t_f, t_\Psi\}$
LSSC failed	$t_f + t_g + t_p$	N / A

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

TABLE II
PER-ITERATION COST OF FISTA, AMGS, AND GENERALIZED ACGM IN THE NON-MONOTONE SETTING

	FISTA	AMGS	ACGM
Base cost	$t_g + t_p$	$2t_g + 2t_p$	$t_g + t_p$
LSSC t_d	t_f	t_g	t_f
LSSC t_c	t_p	$t_g + t_p$	$t_g + t_p$
Backtrack cost	$t_f + t_p$	$2t_g + t_p$	$t_f + t_g + t_p$

IV. EXTRAPOLATED FORM

A. Monotonicity and extrapolation

In the original ACGM [17], the auxiliary point can be obtained from two successive main iterates through extrapolation. Interestingly, this property is preserved for any value of the inverse step size. We show in the following how monotonicity alters this property and bring generalized monotone ACGM to a form in which the auxiliary point is an extrapolation of state variables. First, we observe that estimate sequence vertices can be obtained from main iterates through extrapolation, namely

$$\boldsymbol{v}_{k+1} = \boldsymbol{x}_k + \frac{A_{k+1}}{a_{k+1}}(\boldsymbol{z}_{k+1} - \boldsymbol{x}_k), \quad k \geq 0. \tag{32}$$

The proof does not require any conditions on A_0 or γ_0 and is thus the same as the one in [17, Lemma 5]. Combined with the auxiliary point update (28) it leads to

$$\begin{aligned} \boldsymbol{y}_{k+1} = &\frac{1}{A_k\gamma_{k+1} + a_{k+1}\gamma_k} \\ &\left(A_k\gamma_{k+1}\boldsymbol{x}_k + \frac{A_k}{a_k}\boldsymbol{z}_k + \left(a_{k+1}\gamma_k - \frac{A_k}{a_k}\right)\boldsymbol{x}_{k-1}\right), \end{aligned} \tag{33}$$

for all $k \geq 1$. Depending on the outcome of the update in line 16 of Algorithm 2, we distinguish two situations.

If MC passes at iteration $k-1$ ($F(\boldsymbol{z}_k) \leq F(\boldsymbol{x}_{k-1})$), then

$$\boldsymbol{y}_{k+1} = (1 + b_k)\boldsymbol{z}_k - b_k\boldsymbol{x}_{k-1} = \boldsymbol{x}_k + b_k(\boldsymbol{z}_k - \boldsymbol{x}_{k-1}), \tag{34}$$

where, for brevity, we define extrapolation factor b_k and subexpression ω_k as

$$b_k \stackrel{\text{def}}{=} \left(\frac{A_k}{a_k} - 1\right)\omega_k, \quad \omega_k \stackrel{\text{def}}{=} \frac{a_{k+1}\gamma_k}{A_k\gamma_{k+1} + a_{k+1}\gamma_k}. \tag{35}$$

If the algorithm overshoots ($F(\boldsymbol{z}_k) > F(\boldsymbol{x}_{k-1})$) then, by monotonicity, we impose $\boldsymbol{x}_k = \boldsymbol{x}_{k-1}$, which leads to

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

where the extrapolation factor b'_k is given by

$$b'_k \stackrel{\text{def}}{=} \left(\frac{A_k}{a_k}\right)\omega_k. \tag{37}$$

Expressions (34) and (36) lead to the following auxiliary point extrapolation rule:

$$\boldsymbol{y}_{k+1} = \boldsymbol{x}_k + \beta_k(\boldsymbol{z}_k - \boldsymbol{x}_{k-1}), \quad k \geq 1, \tag{38}$$

where

$$\beta_k = \begin{cases} b_k, & \boldsymbol{x}_k = \boldsymbol{z}_k \\ b'_k, & \boldsymbol{x}_k = \boldsymbol{x}_{k-1} \end{cases}, \quad k \geq 1.$$

Until this point we have assumed that the first iteration $k = 0$ *does not* use auxiliary point extrapolation rule (38). To write generalized ACGM in a form similar to monotone FISTA (MFISTA [18]) and the monotone version of FISTA-CP [6], we define the vertex extrapolation factor in (32) as

$$t_k \stackrel{\text{def}}{=} \frac{A_k}{a_k}, \quad k \geq 1. \tag{39}$$

As long as $k \geq 1$, (30) enables us to obtain a recursion rule for the vertex extrapolation factor that does not depend on weights a_k and A_k, given by

$$t_{k+1}^2 + t_{k+1}(q_k t_k^2 - 1) - \frac{L_{k+1} + \mu_\Psi}{L_k + \mu_\Psi}t_k^2 = 0, \ k \geq 1, \ \mu \geq 0. \tag{40}$$

Subexpression ω_k and auxiliary point extrapolation factor β_k can also be written for all $k \geq 1$ as

$$\omega_k = \frac{1 - q_{k+1}t_{k+1}}{(1 - q_{k+1})t_{k+1}}, \tag{41}$$

$$\beta_k = (t_k - \mathbf{1}_{\{\boldsymbol{z}_k\}}(\boldsymbol{x}_k))\omega_k, \tag{42}$$

where $\mathbf{1}_X$ denotes the membership function of set X, namely

$$\mathbf{1}_X(x) = \begin{cases} 1, & x \in X \\ 0, & x \notin X \end{cases}.$$

Note that subexpression ω_k contains only recent information whereas β_k needs only to access the state of the preceding iteration.

For simplicity, we wish to extend update rules (38), (40), and (42) to the first iteration $k = 0$. The missing parameters follow naturally from this extension. First, t_0 can be obtained by setting $k = 0$ in (40) as

$$\begin{aligned} t_0 = \sqrt{\frac{t_1^2 - t_1}{\frac{L_1 + \mu_\Psi}{L_0 + \mu_\Psi} - t_1 q_0}} &\stackrel{(39)}{=} \sqrt{\frac{A_1 - a_1}{\frac{(L_1 + \mu_\Psi)a_1^2}{(L_0 + \mu_\Psi)A_1} - a_1 q_0}} \\ &\stackrel{(30)}{=} \sqrt{\frac{(L_0 + \mu_\Psi)A_0}{\gamma_0}}. \end{aligned} \tag{43}$$

Next, we introduce a "phantom iteration" $k = -1$ with the main iterate as the only state parameter. We set $x_{-1} \stackrel{\text{def}}{=} x_0$ so that *any* value of β_0 will satisfy (38). For brevity, we obtain β_0 from expression (42) with $k = 0$. We do not define a_0. Instead, extrapolation factors b_k and b'_k from (35) and (37)

can be computed when $k = 0$ by replacing A_0/a_0 with the t_0 expression in (46).

Thus, with initialization (46) and recursion (40), we have completely defined the vertex extrapolation factor sequence $\{t_k\}_{k\geq 0}$, and derived from it the auxiliary extrapolation factor expression (42). Now, we do not need to maintain weight sequences $\{a_k\}_{k\geq 1}$ and $\{A_k\}_{k\geq 0}$. We simplify generalized ACGM further by noting that, to produce the auxiliary point, extrapolation rule (38) depends on three vector parameters. However, it is not necessary to store both $\boldsymbol{z}_k$ and $\boldsymbol{x}_{k-1}$ across iterations. To address applications where memory is limited, we only maintain the difference term $\boldsymbol{d}_k$, given by

$$\boldsymbol{d}_k = (t_k - \mathbf{1}_{\{\boldsymbol{z}_k\}}(\boldsymbol{x}_k))(\boldsymbol{z}_k - \boldsymbol{x}_{k-1}), \quad k \geq 0. \tag{44}$$

Extrapolation rule (38) becomes

$$\boldsymbol{y}_{k+1} = \boldsymbol{x}_k + \omega_k \boldsymbol{d}_k, \quad k \geq 0. \tag{45}$$

The above modifications yield a form of generalized ACGM based on extrapolation, which we list in Algorithm 3. To obtain a non-monotone algorithm, it suffices to replace line 16 of Algorithm 3 with (31).

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

Algorithm 3 Generalized monotone ACGM in extrapolated form

ACGM($\boldsymbol{x_0}$, L_0, μ_f, μ_Ψ, A_0, γ_0, r_u, r_d, K)

1: $\boldsymbol{x}_{-1} = \boldsymbol{x}_0,\ \boldsymbol{d}_0 = \mathbf{0}$
2: $\mu = \mu_f + \mu_\Psi,\ t_0 = \sqrt{\frac{(L_0+\mu_\Psi)A_0}{\gamma_0}},\ q_0 = \frac{\mu}{L_0+\mu_\Psi}$
3: **for** $k = 0, ..., K-1$ **do**
4: $\quad \hat{L}_{k+1} := r_d L_k$
5: $\quad$ **loop**
6: $\quad\quad \hat{q}_{k+1} := \frac{\mu}{\hat{L}_{k+1}+\mu_\Psi}$
7: $\quad\quad \hat{t}_{k+1} := \frac{1}{2}\left(1 - q_k t_k^2 + \sqrt{(1-q_k t_k^2)^2 + 4\frac{\hat{L}_{k+1}+\mu_\Psi}{L_k+\mu_\Psi}t_k^2}\right)$
8: $\quad\quad \hat{\boldsymbol{y}}_{k+1} := \boldsymbol{x}_k + \frac{1-\hat{q}_{k+1}\hat{t}_{k+1}}{(1-\hat{q}_{k+1})\hat{t}_{k+1}}\boldsymbol{d}_k$
9: $\quad\quad \hat{\boldsymbol{z}}_{k+1} := \mathrm{prox}_{\frac{1}{\hat{L}_{k+1}}\Psi}\left(\hat{\boldsymbol{y}}_{k+1} - \frac{1}{\hat{L}_{k+1}}\nabla f(\hat{\boldsymbol{y}}_{k+1})\right)$
10: $\quad\quad$ **if** $f(\hat{\boldsymbol{z}}_{k+1}) \leq Q_{f,\hat{L}_{k+1},\hat{\boldsymbol{y}}_{k+1}}(\hat{\boldsymbol{z}}_{k+1})$ **then**
11: $\quad\quad\quad$ Break from loop
12: $\quad\quad$ **else**
13: $\quad\quad\quad \hat{L}_{k+1} := r_u \hat{L}_{k+1}$
14: $\quad\quad$ **end if**
15: $\quad$ **end loop**
16: $\quad \boldsymbol{x}_{k+1} := \arg\min\{F(\hat{\boldsymbol{z}}_{k+1}), F(\boldsymbol{x}_k)\}$
17: $\quad \boldsymbol{d}_{k+1} := (\hat{t}_{k+1} - \mathbf{1}_{\{\hat{\boldsymbol{z}}_{k+1}\}}(\boldsymbol{x}_{k+1}))(\hat{\boldsymbol{z}}_{k+1} - \boldsymbol{x}_k)$
18: $\quad L_{k+1} := \hat{L}_{k+1},\ q_{k+1} := \hat{q}_{k+1},\ t_{k+1} := \hat{t}_{k+1}$
19: **end for**
20: **return** x_K

B. Retrieving the convergence guarantee

For Algorithm 2, the convergence guarantee in (3) is obtained directly from the state variable A_k. For Algorithm 3, we need the following result.

Lemma 2. *The vertex extrapolation factor expression in* (46) *generalizes to arbitrary $k \geq 0$ as*

$$t_k = \sqrt{\frac{(L_k + \mu_\Psi)A_k}{\gamma_k}}.$$

Proof: For $k = 0$, (46) holds. For $k \geq 1$ we have that

$$t_{k+1} = \sqrt{\frac{(L_{k+1}+\mu_\Psi)A_{k+1}^2}{(L_{k+1}+\mu_\Psi)a_{k+1}^2}} \overset{(30)}{=} \sqrt{\frac{(L_{k+1}+\mu_\Psi)A_{k+1}}{\gamma_{k+1}}}.$$

■

From Lemma 2, we distinguish two scenarios.

If $\gamma_0 \neq A_0\mu$, the convergence guarantee can be derived directly from the state parameters without alterations to Algorithm 3 as

$$A_k = \frac{(\gamma_0 - A_0\mu)t_k^2}{(L_k+\mu_\Psi)(1-q_k t_k^2)}, \quad k \geq 1.$$

C. Border-case

However, if $\gamma_0 = A_0\mu$, then

$$t_k = \frac{1}{\sqrt{q_k}}, \quad k \geq 1. \tag{46}$$

Therefore, the state parameters of Algorithm 3 no longer contain information on the convergence guarantee but can be brought to a simpler form. The result in (46) implies that the auxiliary point extrapolation factor is given by

$$\beta_k = \frac{\sqrt{L_k+\mu_\Psi} - \mathbf{1}_{\{\boldsymbol{z}_k\}}(\boldsymbol{x}_k)\sqrt{\mu}}{\sqrt{L_{k+1}+\mu_\Psi} + \sqrt{\mu}}, \quad k \geq 0. \tag{47}$$

The sequence $\{t_k\}_{k\geq 0}$ does not store any relevant information and can be left out. This means that the convergence guarantee A_k requires a dedicated update. A simple recursion rule follows from (46) as

$$A_{k+1} = \frac{1}{1-\sqrt{q_{k+1}}}A_k, \quad k \geq 0. \tag{48}$$

Due to scaling invariance, we can select any pair (A_0, γ_0) that is a positive multiple of $(1, \mu)$. For simplicity, we choose $A_0 = 1$ and $\gamma_0 = \mu$.

To reduce computational intensity, we modify subexpressions $\boldsymbol{d}_k$ and ω_k as

$$\boldsymbol{d}_k = \left(\sqrt{L_k+\mu_\Psi} - \mathbf{1}_{\{\boldsymbol{z}_k\}}(\boldsymbol{x}_k)\sqrt{\mu}\right)(\boldsymbol{z}_k - \boldsymbol{x}_{k-1}), \quad k \geq 0, \tag{49}$$

$$\omega_k = \frac{1}{\sqrt{L_{k+1}+\mu_\Psi} + \sqrt{\mu}}, \quad k \geq 0. \tag{50}$$

The local inverse condition number sequence $\{q_k\}_{k\geq 0}$ does not appear in updates (47) and (48). Hence, it can also be abstracted away. The form taken by generalized ACGM in this border-case, after simplifications, is listed in Algorithm 4.

A non-monotone variant can be obtained by replacing line 13 of Algorithm 4, with (31). The border-case iteration complexity matches the one of Algorithms 2 and 3.

Algorithm 4 Border-case ACGM in extrapolated form
ACGM($\boldsymbol{x_0}$, L_0, μ_f, μ_Ψ, r_u, r_d, K)

$\boldsymbol{x}_{-1} = \boldsymbol{x}_0,\ \boldsymbol{d}_0 = \boldsymbol{0},\ A_0 = 1,\ \mu = \mu_f + \mu_\Psi$
for $k = 0, ..., K-1$ **do**
  $\hat{L}_{k+1} := r_d L_k$
  loop
    $\hat{\boldsymbol{y}}_{k+1} := \boldsymbol{x}_k + \frac{1}{\sqrt{\hat{L}_{k+1}+\mu_\Psi}+\sqrt{\mu}}\boldsymbol{d}_k$
    $\hat{\boldsymbol{z}}_{k+1} := \text{prox}_{\frac{1}{\hat{L}_{k+1}}\Psi}\left(\hat{\boldsymbol{y}}_{k+1} - \frac{1}{\hat{L}_{k+1}}\nabla f(\hat{\boldsymbol{y}}_{k+1})\right)$
    if $f(\hat{\boldsymbol{z}}_{k+1}) \le Q_{f,\hat{L}_{k+1},\hat{\boldsymbol{y}}_{k+1}}(\hat{\boldsymbol{z}}_{k+1})$ **then**
      Break from loop
    else
      $\hat{L}_{k+1} := r_u \hat{L}_{k+1}$
    end if
  end loop
  $\boldsymbol{x}_{k+1} := \arg\min\{F(\hat{\boldsymbol{z}}_{k+1}), F(\boldsymbol{x}_k)\}$
  $\boldsymbol{d}_{k+1} := \left(\sqrt{\hat{L}_{k+1}+\mu_\Psi} - \boldsymbol{1}_{\{\hat{\boldsymbol{z}}_{k+1}\}}(\boldsymbol{x}_{k+1})\sqrt{\mu}\right)(\hat{\boldsymbol{z}}_{k+1} - \boldsymbol{x}_k)$
  $L_{k+1} := \hat{L}_{k+1}$
  $A_{k+1} := \frac{\sqrt{\hat{L}_{k+1}+\mu_\Psi}}{\sqrt{\hat{L}_{k+1}+\mu_\Psi}-\sqrt{\mu}}A_k$
end for
return x_K

V. Simulation results

A. Benchmark setup

We have tested the variants of generalized ACGM introduced in this work against the methods considered at the time of writing to be the state-of-the-art on the problem class outlined in Subsection I-A. The proposed methods included in the benchmark are non-monotone ACGM (denoted as plain ACGM), monotone ACGM (MACGM), and, for strongly-convex problems, border-case non-monotone ACGM (BACGM) as well as border-case monotone ACGM (BMACGM). The state-of-the-art methods are FISTA-CP, monotone FISTA-CP (MFISTA-CP) [6], AMGS [9], and FISTA with backtracking line-search (FISTA-BT) [10].

We have selected as test cases five synthetic instances of composite problems in the areas of statistics, inverse problems, and machine learning. Three are non-strongly convex: least absolute shrinkage and selection operator (LASSO) [11], non-negative least squares (NNLS), and l_1-regularized logistic regression (L1LR). The other two are strongly-convex: ridge regression (RR) and elastic net (EN) [22]. Table III lists the oracle functions of all above mentioned problems.

Here, the sum softplus function $\mathcal{I}(\boldsymbol{x})$, the element-wise logistic function $\mathcal{L}(\boldsymbol{x})$, and the shrinkage operator $\mathcal{T}_\tau(\boldsymbol{x})$ are, respectively, given by

$$\mathcal{I}(\boldsymbol{x}) = \sum_{i=1}^{m} \log(1 + e^{\boldsymbol{x}_i}), \quad i \in \{1, ..., m\},$$

$$\mathcal{L}(\boldsymbol{x})_i = \frac{1}{1 + e^{-\boldsymbol{x}_i}}, \quad i \in \{1, ..., m\},$$

$$\mathcal{T}_\tau(\boldsymbol{x})_j = (|\boldsymbol{x}_j| - \tau)_+ \operatorname{sgn}(\boldsymbol{x}_j), \quad j \in \{1, ..., n\}.$$

To attain the best convergence guarantees for AMGS, Nesterov suggests in [9] that all known global strong convexity be transfered to the simple function Ψ. When line-search is enabled, generalized ACGM also benefits slightly from this arrangement when $r_u > 1$ (Theorem 2). Without line-search, the convergence guarantees of generalized ACGM do not change as a result of strong convexity transfer, in either direction. Thus, for fair comparison, we have incorporated in Ψ the strongly-convex quadratic regularization term for RR and EN problems. In the following, we describe in detail each of the five problem instances. All random variables are independent and identically distributed, unless stated otherwise.

LASSO. Real-valued matrix $\boldsymbol{A}$ is of size $m = 500$ by $n = 500$, with entries drawn from $\mathcal{N}(0, 1)$. Vector $\boldsymbol{b} \in \mathbb{R}^m$ has entries sampled from $\mathcal{N}(0, 9)$. Regularization parameter λ_1 is 4. The starting point $\boldsymbol{x}_0 \in \mathbb{R}^n$ has entries drawn from $\mathcal{N}(0, 1)$.

NNLS. Sparse $m = 1000 \times n = 10000$ matrix $\boldsymbol{A}$ has approximately 10% of entries, at random locations, non-zero. The non-zero entries are drawn from $\mathcal{N}(0, 1)$ after which each column $j \in \{1, ..., n\}$ is scaled independently to have an l_2 norm of 1. Starting point $\boldsymbol{x}_0$ has 10 entries at random locations all equal to 4 and the remainder zero. Vector $\boldsymbol{b}$ is obtained from $\boldsymbol{b} = \boldsymbol{A}\boldsymbol{x}_0 + \boldsymbol{z}$, where $\boldsymbol{z}$ is standard Gaussian noise.

L1LR. Matrix $\boldsymbol{A}$ has $m = 200 \times n = 1000$ entries sampled from $\mathcal{N}(0, 1)$, $\boldsymbol{x}_0$ has exactly 10 non-zero entries at random locations, each entry value drawn from $\mathcal{N}(0, 225)$, and $\lambda_1 = 5$. Labels $\boldsymbol{y}_i \in \{0, 1\}, i \in \{1, ..., m\}$ are selected with probability $\mathbb{P}(\boldsymbol{Y}_i = 1) = \mathcal{L}(\boldsymbol{A}\boldsymbol{x})_i$.

RR. Dimensions are $m = 500 \times n = 500$. The entries of matrix $\boldsymbol{A}$, vector $\boldsymbol{b}$, and starting point $\boldsymbol{x}_0$ are drawn from $\mathcal{N}(0, 1)$, $\mathcal{N}(0, 25)$, and $\mathcal{N}(0, 1)$, respectively. Regularizer λ_2 is given by $10^{-3}(\sigma_{max}(\boldsymbol{A}))^2$, where $\sigma_{max}(\boldsymbol{A})$ is the largest singular value of $\boldsymbol{A}$.

EN. Matrix $\boldsymbol{A}$ has $m = 1000 \times n = 500$ entries sampled from $\mathcal{N}(0, 1)$. Starting point $\boldsymbol{x}_0$ has 20 non-zero entries at random locations, each entry value drawn from $\mathcal{N}(0, 1)$. Regularization parameter λ_1 is obtained according to [23] as $\lambda_1 = 1.5\sqrt{2\log(n)}$ and λ_2 is the same as in RR, $\lambda_2 = 10^{-3}(\sigma_{max}(\boldsymbol{A}))^2$.

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

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

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

The line-search parameters were selected according to the recommendation given in [4]. For AMGS and FISTA-BT we have $r_u^{\text{AMGS}} = r_u^{\text{FISTA}} = 2.0$ and $r_d^{\text{AMGS}} = 0.9$.

TABLE III
ORACLE FUNCTIONS OF THE FIVE TEST PROBLEMS

	$f(\boldsymbol{x})$	$\Psi(\boldsymbol{x})$	$\nabla f(\boldsymbol{x})$	$prox_{\tau\Psi}(\boldsymbol{x})$
LASSO	$\frac{1}{2}\|\boldsymbol{Ax}-\boldsymbol{b}\|_2^2$	$\lambda_1\|\boldsymbol{x}\|_1$	$\boldsymbol{A}^T(\boldsymbol{Ax}-\boldsymbol{b})$	$\mathcal{T}_{\tau\lambda_1}(\boldsymbol{x})$
NNLS	$\frac{1}{2}\|\boldsymbol{Ax}-\boldsymbol{b}\|_2^2$	$\sigma_{\mathbb{R}^n_+}(\boldsymbol{x})$	$\boldsymbol{A}^T(\boldsymbol{Ax}-\boldsymbol{b})$	$(\boldsymbol{x})_+$
L1LR	$\mathcal{I}(\boldsymbol{Ax})-\boldsymbol{y}^T\boldsymbol{Ax}$	$\lambda_1\|\boldsymbol{x}\|_1$	$\boldsymbol{A}^T(\mathcal{L}(\boldsymbol{Ax})-\boldsymbol{y})$	$\mathcal{T}_{\tau\lambda_1}(\boldsymbol{x})$
RR	$\frac{1}{2}\|\boldsymbol{Ax}-\boldsymbol{b}\|_2^2$	$\frac{\lambda_2}{2}\|\boldsymbol{x}\|_2^2$	$\boldsymbol{A}^T(\boldsymbol{Ax}-\boldsymbol{b})$	$\frac{1}{1+\tau\lambda_2}\boldsymbol{x}$
EN	$\frac{1}{2}\|\boldsymbol{Ax}-\boldsymbol{b}\|_2^2$	$\lambda_1\|\boldsymbol{x}\|_1+\frac{\lambda_2}{2}\|\boldsymbol{x}\|_2^2$	$\boldsymbol{A}^T(\boldsymbol{Ax}-\boldsymbol{b})$	$\frac{1}{1+\tau\lambda_2}\mathcal{T}_{\tau\lambda_1}(\boldsymbol{x})$

The variants of generalized ACGM and AMGS are the only methods included in the benchmark that are equipped with fully adaptive line-search. We have decided to select r_d^{ACGM} to ensure that ACGM and AMGS have the same overhead. We formally define the line-search overhead of method $\mathcal{M}$, denoted by $\Omega^{\mathcal{M}}$, as the average computational cost attributable to backtracks per WTU of advancement. Assuming that the LCEs hover around a fixed value (Subsection III-C), we thus have that

$$\Omega^{\mathrm{AMGS}} = -\frac{(2t_g+t_p)\log(r_d^{\mathrm{AMGS}})}{2(t_g+t_p)\log(r_u^{\mathrm{AMGS}})}, \tag{51}$$

$$\Omega^{\mathrm{ACGM}} = -\frac{(t_f+t_g+t_p)\log(r_d^{\mathrm{ACGM}})}{(t_g+t_p)\log(r_u^{\mathrm{ACGM}})}. \tag{52}$$

From (51) and (52) we have that $r_d^{\mathrm{ACGM}} = (r_d^{\mathrm{AMGS}})^{\frac{2}{3}}$, with no difference for border-case or monotone variants.

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

B. Non-strongly convex problems

The convergence results for LASSO, NNLS, and L1LR are shown in Figure 1. The LCE variation during the first 200 WTU is shown in Figure 2. For NNLS, floating point precision was exhausted after 100 WTU and the LCE variation was only plotted to this point (Figure 2(b)). In addition, the average LCEs are listed in Table IV.

Both variants of ACGM outperform in iterations and especially in WTU the competing methods in each of these problem instances. Even though for LASSO and NNLS, the iteration convergence rate of AMGS is slightly better in the beginning (Figures 1(a) and 1(c)), AMGS lags behind afterwards and, when measured in terms of computational convergence rate, has the poorest performance among the methods tested (Figures 1(b), 1(d), and 1(f)). FISTA-BT produces the same iterates as FISTA-CP, as theoretically guaranteed in the non-strongly convex case for $L_0 = L_f$.

The overall superiority of ACGM and MACGM can be attributed to the effectiveness of line-search. Interestingly, ACGM manages to surpass FISTA-CP and MFISTA-CP even when the latter are supplied with the exact value of the global Lipschitz constant. This is because ACGM is able to accurately estimate the local curvature, which is often below L_f. For the L1LR problem, where the smooth part f is not the square of a linear function, the local curvature is substantially lower than the global Lipschitz constant with LCEs hovering around one fifth of L_f (Figure 2(c)). One would expect AMGS to be able to estimate local curvature as accurately as ACGM. This is does not happen due to AMGS's reliance on a "damped relaxation condition" [9] line-search stopping criterion. For LASSO and NNLS, the average LCE of AMGS is actually above L_f. ACGM has an average LCE that is roughly two thirds that of AMGS on these problems whereas for L1LR the average is more than three times lower than AMGS. The difference between the LCE averages of ACGM and MACGM is negligible.

Indeed, monotonicity, as predicted, does not alter the overall iteration convergence rate and has a stabilizing effect. MACGM overshoots do have a negative but limited impact on the computational convergence rate. We have noticed in our simulations that overshoots occur less often for larger problems, such as the tested instance of NNLS.

C. Strongly convex problems

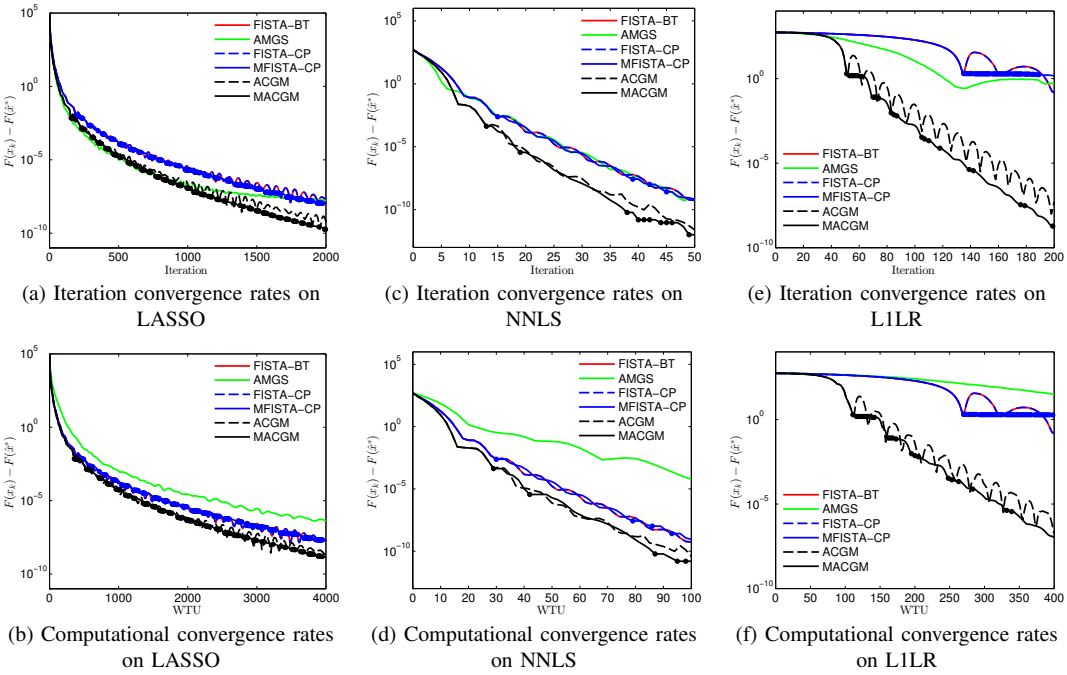
The convergence results for RR and EN are shown in Figures 3(a), 3(b), 3(c), and 3(d). The LCE variation is shown in Figure 4 with LCE averages listed in Table V.

Strongly convex problems have a unique optimum point and accelerated first-order schemes are guaranteed to find an accurate estimate of it in domain space (see [13] for detailed analysis). Along with Theorem 2, it follows that

$$A_0(F(\boldsymbol{x}_0)-F(\hat{\boldsymbol{x}}^*))+\frac{\gamma_0}{2}\|\boldsymbol{x}_0-\hat{\boldsymbol{x}}^*\|_2^2 \simeq \Delta_0.$$

Thus, we can display accurate estimates of ISDUBs in (3), of the form $U_k \overset{\mathrm{def}}{=} \Delta_0/A_k$, $k \geq 1$, for methods that maintain convergence guarantees at runtime. These are shown in Figures 3(e) and 3(f) as upper bounds indexed in WTU.

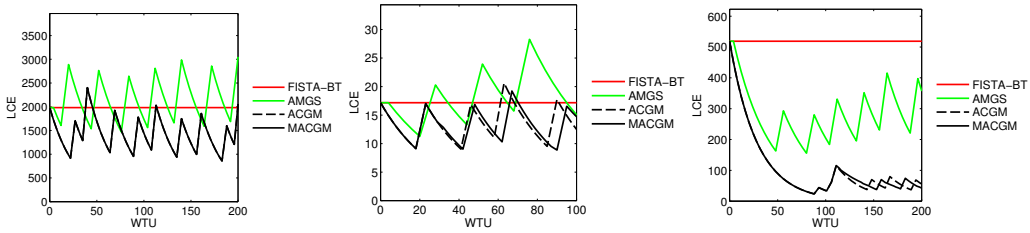
For the smooth RR problem, the effectiveness of each algorithm tested is roughly given by the increase rate of the accumulated weights (Figures 3(a) and 3(c)). In iterations, AMGS converges the fastest. However, in terms of WTU usage, it is the least effective of the methods designed to deal with strongly convex objectives. The reasons are the high cost of its iterations, its low asymptotic rate compared to ACGM and FISTA-CP, and the stringency of its damped relaxation criterion that results in higher LCEs (on average) than ACGM (Figure 4(a) and Table V). The computational convergence rate of BACGM is the best, followed by ACGM, FISTA-CP. This does not, however, correspond to the upper bounds (Figure 3(e)). While BACGM produces the largest



(a) Iteration convergence rates on LASSO (c) Iteration convergence rates on NNLS (e) Iteration convergence rates on L1LR

(b) Computational convergence rates on LASSO (d) Computational convergence rates on NNLS (f) Computational convergence rates on L1LR

Fig. 1. Convergence results of FISTA with backtracking (FISTA-BT), AMGS, FISTA-CP, monotone FISTA-CP (MFISTA-CP), non-monotone ACGM and monotone ACGM (MACGM) on the LASSO, NNLS, and L1LR non-strongly convex problems. Dots mark iterations *preceding* overshoots. At these iterations, the convergence behavior changes.



(a) LASSO (b) NNLS (c) L1LR

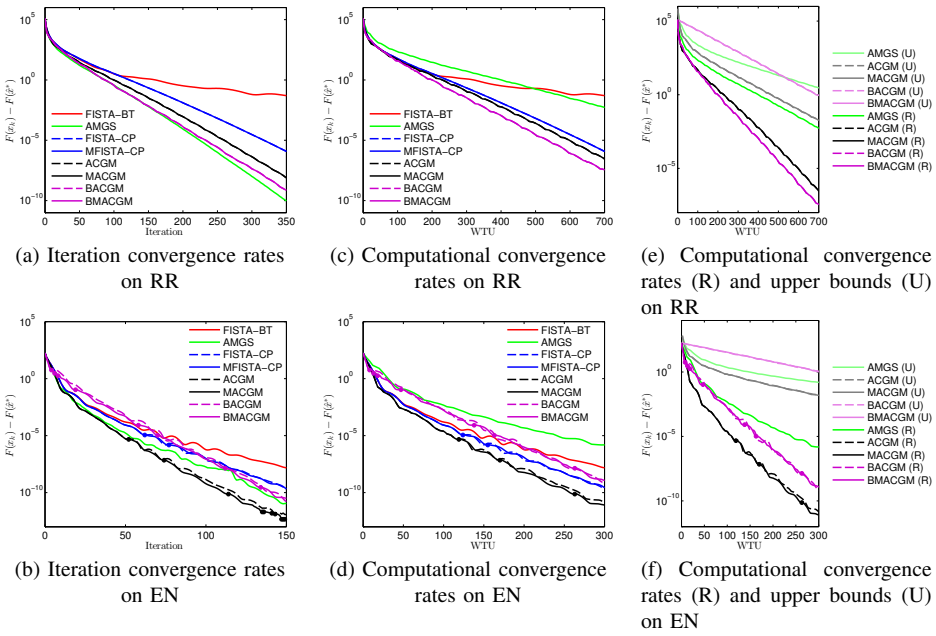
Fig. 2. Line-search method LCE variation on LASSO, NNLS, and L1LR

TABLE IV
AVERAGE LCES OF LINE-SEARCH METHODS ON LASSO, NNLS, AND L1LR

Problem	L_f	Iterations	FISTA-BT	AMGS	ACGM	MACGM
LASSO	1981.98	2000	1981.98	2202.66	1385.85	1303.70
NNLS	17.17	50	17.17	19.86	14.35	13.54
L1LR	518.79	200	518.79	246.56	80.76	79.12

TABLE V
AVERAGE LCES OF LINE-SEARCH METHODS ON RR AND EN

Problem	L_f	Iterations	FISTA-BT	AMGS	ACGM	MACGM	BACGM	BMACGM
RR	1963.66	350	1963.66	2022.73	1473.88	1473.88	1471.16	1471.16
EE	2846.02	150	2846.02	3023.47	2056.68	2003.09	2093.56	1998.12



(a) Iteration convergence rates on RR

(c) Computational convergence rates on RR

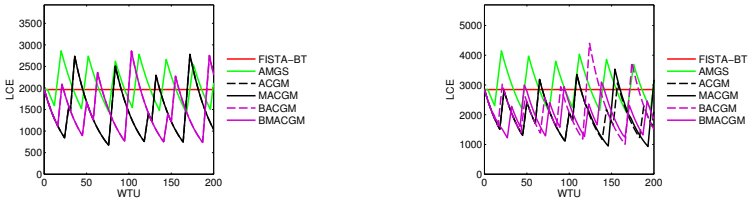
(e) Computational convergence rates (R) and upper bounds (U) on RR

(b) Iteration convergence rates on EN

(d) Computational convergence rates on EN

(f) Computational convergence rates (R) and upper bounds (U) on EN

Fig. 3. Convergence results of FISTA with backtracking (FISTA-BT), AMGS, FISTA-CP, monotone FISTA-CP (MFISTA-CP), non-monotone ACGM, monotone ACGM (MACGM), border-case non-monotone ACGM (BACGM), and border-case monotone ACGM (BMACGM) on the RR and EN strongly-convex problems. Dots mark iterations preceding overshoots.



(a) RR

(b) EN

Fig. 4. Line-search method LCE variation on RR and EN

accumulated weights A_k, the high value of the ISD term in Δ_0 causes BACGM to have poorer upper bounds than ACGM, except for the first iterations. In fact, the effectiveness of BACGM on this problem is exceptional, partly due to the regularity of the composite gradients. This regularity also ensures monotonicity of BACGM, ACGM, and FISTA-CP. FISTA-BT does not exhibit linear convergence on this problem and it is even slower than AMGS after 500 WTU despite its lower line-search overhead and advantageous parameter choice $L_0 = L_f$.

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

VI. Discussion

Our simulation results suggest that enforcing monotonicity in ACGM is generally beneficial in large-scale applications. It leads to a more predictable convergence rate and, provided that the number of overshoots per iteration is small, monotonicity has a negligible impact on the computational convergence rate as well. Our experimental results indicate that the frequency of overshoots generally decreases with problem size.

From a theoretical standpoint, the proposed method can be viewed as a unification of FGM and FISTA, in their most common forms. Specifically, the fixed-step variant ($L_k = L_f$ for all $k \geq 0$) of ACGM in extrapolated form (Algorithm 3) is equivalent to both the monotone and non-monotone variants of FISTA-CP with the theoretically optimal step size

TABLE VI
FGM AND FISTA, ALONG WITH THEIR COMMON VARIANTS, CAN BE CONSIDERED INSTANCES OF GENERALIZED ACGM WITH CERTAIN RESTRICTIONS APPLIED.

Algorithm	Restriction						
	Smooth objective	$\mu = 0$	$\mu > 0$	$A_0 = 0$	$A_0 > 0$	Fixed step size	Non-monotone
FGM [13]	yes	no	no	no	yes	yes	yes
FGM [14]	yes	yes	no	unclear	unclear	no	yes
FISTA [10]	no	yes	no	yes	no	partial	yes
MFISTA [18]	no	yes	no	yes	no	yes	no
scAPG [12]	no	no	yes	no	yes	no	yes
FISTA-CP [6]	no	no	no	no	no	yes	no

$\tau^{\mathrm{FISTA-CP}} = \frac{1}{L_f}$. Moreover, when $\mu = 0$, non-monotone original fixed-step ACGM coincides with the original formulation of FISTA in [10]. Adding monotonicity yields MFISTA [18]. Also for $\mu = 0$, but without the fixed-step restriction, the original non-monotone ACGM in estimate sequence form reduces to the robust FISTA-like algorithm in [16], whereas in extrapolated form it is a valuable simplification of the method introduced in [15].

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

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

where $\boldsymbol{x}_{k+1}$ is given by line 5 of Algorithm 1, we obtain the "general scheme of optimal method" in [13]. Both the monotone and non-monotone variants adhere to this scheme. The correspondence between Nesterov's notation in [13] and ours is, for all $k \geq 0$, given by:

$$\begin{aligned}
\lambda_k^{\mathrm{FGM}} &= \frac{A_0^{\mathrm{ACGM}}}{A_k^{\mathrm{ACGM}}},\\
\phi_k^{\mathrm{FGM}}(\boldsymbol{x}) &= \frac{1}{A_k^{\mathrm{ACGM}}}\psi_k^{\mathrm{ACGM}}(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^n\\
\boldsymbol{y}_k^{\mathrm{FGM}} &= \boldsymbol{y}_{k+1}^{\mathrm{ACGM}},\\
\alpha_k^{\mathrm{FGM}} &= \frac{a_{k+1}^{\mathrm{ACGM}}}{A_{k+1}^{\mathrm{ACGM}}} = \frac{1}{t_{k+1}^{\mathrm{ACGM}}},\\
\gamma_k^{\mathrm{FGM}} &= \frac{\gamma_k^{\mathrm{ACGM}}}{A_k^{\mathrm{ACGM}}}.
\end{aligned}$$

The remaining state parameters are identical. Note that FGM makes the assumption that $A_0^{\mathrm{ACGM}} > 0$, which is incompatible with the original specification of ACGM in [17]. With the above assumption, the non-monotone variant (Algorithm 2) is in fact identical to "constant step scheme I" in [13]. Similarly, the extrapolated form of fixed-step non-monotone ACGM (Algorithm 3) corresponds exactly to "constant step scheme II" in [13] while fixed-step border-case non-monotone ACGM (Algorithm 4) is identical to "constant step scheme III" in [13]. We note that for both the RR and EN problems, regardless of the actual performance of BACGM, the convergence guarantees of BACGM are poorer than those of ACGM with $A_0 = 0$. This discrepancy in guarantees is supported theoretically because, in the most common applications, the ISD term in Δ_0 is large compared to the DST. This extends to the fixed-step setup and challenges the notion found in the literature (e.g., [24]) that for strongly-convex functions, FGM and FISTA-CP are momentum methods that take the form of the "constant step scheme III" in [13]. In fact, the border-case form may constitute the poorest choice of parameters A_0 and γ_0 in many applications. Indeed, the worst-case results in Theorem 2 favor $A_0 = 0$.

The FGM variant in [14] is a particular case of non-monotone ACGM with variable step size (Algorithm 2) when the objective is non-strongly convex ($\mu = 0$) and the step size search parameters are set to $r_u^{\mathrm{ACGM}} = 2$ and $r_d^{\mathrm{ACGM}} = 0.5$. The notation correspondence is as follows:

$$\begin{aligned}
\boldsymbol{x}_{k+1,i}^{\mathrm{FGM}} &= \boldsymbol{x}_{k+1}^{\mathrm{ACGM}}, \quad \boldsymbol{y}_{k,i}^{\mathrm{FGM}} = \boldsymbol{y}_{k+1}^{\mathrm{ACGM}},\\
a_{k,i}^{\mathrm{FGM}} &= \hat{a}_{k+1}^{\mathrm{ACGM}}, \quad 2^i L_f = \hat{L}_{k+1}^{\mathrm{ACGM}}.
\end{aligned}$$

The remaining parameters are identical.

Thus, by relaxing the assumption that $A_0 = 0$, we have devised a generalized variant of ACGM that effectively encompasses FGM [13], with its recently introduced variant [14], as well as the original FISTA [10], including its adaptive step-size variants [15], [16], the monotone version MFISTA [18], and the strongly convex extension FISTA-CP [6]. A summary of how the above first-order methods relate to generalized ACGM is given in Table VI.

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

In terms of usability, generalized ACGM does not require a priori knowledge of Lipschitz constant L_f, or a lower estimate of it, beforehand. Thus, the proposed method can be utilized *without any quantitative knowledge of the problem*. Lack of information does not hinder generalized ACGM more than any other primal first-order method while additional information, such as values of strong convexity parameters μ_f and μ_Ψ or even an accurate estimate L_0 of the curvature around $\boldsymbol{x}_0$, leads to a state-of-the-art performance increase unsurpassed for its class.

APPENDIX A
PROOF OF THEOREM 1

We assume $k \geq 0$ throughout this proof. The descent condition assumption implies lower bound property (10). Let the tightness of this lower bound be given by the residual $R(\boldsymbol{x})$ as

$$R(\boldsymbol{x}) \stackrel{\text{def}}{=} F(\boldsymbol{x}) - \mathcal{R}_{L_{k+1}+\mu_\Psi, \boldsymbol{y}_{k+1}}(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^n.$$

where $w_{k+1}(\boldsymbol{x})$ is given by (10). From (13) and (19) we have that $\gamma_{k+1} \geq \gamma_k$. Along with $A_k R(\boldsymbol{x}_k) + a_{k+1} R(\boldsymbol{x}^*) \geq 0$, we obtain, using the proof mechanics of [17, Theorem 1], that

$$\begin{aligned} &A_k(F(\boldsymbol{x}_k) - F(\boldsymbol{x}^*)) - A_{k+1}(F(\boldsymbol{z}_{k+1}) - F(\boldsymbol{x}^*)) \geq \\ &\geq \frac{\gamma_{k+1}}{2}\|\boldsymbol{v}_{k+1} - \boldsymbol{x}^*\|_2^2 - \frac{\gamma_k}{2}\|\boldsymbol{v}_k - \boldsymbol{x}^*\|_2^2 + \mathcal{A}_{k+1} + \mathcal{B}_{k+1}. \end{aligned} \tag{53}$$

Combining $F(\boldsymbol{x}_{k+1}) \leq F(\boldsymbol{z}_{k+1})$ with (53) gives the desired result.

APPENDIX B
PROOF OF THEOREM 2

The non-negativity of the weights (13) implies that $\gamma_k \geq \gamma_0$ for all $k \geq 0$. Combined with (30), we have

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

As we can see from Algorithm 2, scaling A_0 and γ_0 by a fixed factor does not alter the behavior of generalized ACGM. Additionally, γ_0 is guaranteed to be non-zero. To simplify calculations, we introduce the normalized convergence guarantees $\bar{A}_k \stackrel{\text{def}}{=} A_k/\gamma_0$ for all $k \geq 0$.

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

$$\bar{A}_{k+1} \geq \bar{A}_k + \frac{1}{2(L_u - \mu_f)} + \sqrt{\frac{1}{4(L_u - \mu_f)^2} + \frac{\bar{A}_k}{(L_u - \mu_f)}}$$

for all $k \geq 0$. Taking into account that $A_0 \geq 0$, we obtain by induction that

$$\bar{A}_k \geq \frac{(k+1)^2}{4(L_u - \mu_f)}, \quad k \geq 1. \tag{54}$$

From assumption $\gamma_0 \geq A_0 \mu$, (21) implies $\gamma_k \geq A_k \mu$ for all $k \geq 0$. Hence

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

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

$$\bar{A}_k \geq \frac{1}{L_u - \mu_f}(1 - \sqrt{q_u})^{-(k-1)}, \quad k \geq 1. \tag{55}$$

Substituting (54) and (55) in (3) completes the proof.

REFERENCES

[1] A. Nemirovski and D.-B. Yudin, *Problem complexity and method efficiency in optimization*. New York, NY: John Wiley & Sons, 1983.

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

[3] ——, "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.

[4] 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.

[5] S. Bubeck *et al.*, "Convex optimization: Algorithms and complexity," *Found. Trends Mach. Learn.*, vol. 8, no. 3-4, pp. 231–357, 2015.

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

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

[8] 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, Sept. 2014.

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

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

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

[12] Q. Lin and L. Xiao, "An adaptive accelerated proximal gradient method and its homotopy continuation for sparse optimization," in *ICML*, 2014, pp. 73–81.

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

[14] Y. Nesterov and S. Stich, "Efficiency of accelerated coordinate descent method on structured optimization problems," Université catholique de Louvain, Tech. Rep. 03, 2016.

[15] 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.

[16] M. I. Florea and S. A. Vorobyov, "A robust FISTA-like algorithm," in *Proc. of IEEE Intern. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, March 2017, New Orleans, USA, pp. 4521–4525.

[17] ——, "An accelerated composite gradient method for large-scale composite objective problems," *IEEE Transactions on Signal Processing (accepted)*, May 2018.

[18] A. Beck and M. Teboulle, "Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems," vol. 18, no. 11, pp. 2419–2434, 2009.

[19] 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.

[20] R. T. Rockafellar, *Convex Analysis*. Princeton University Press, Princeton, NJ, 1970.

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

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

[23] T. Hastie, R. Tibshirani, and M. Wainwright, *Statistical Learning with Sparsity: The Lasso and Generalizations*. CRC Press, 2015.

[24] W. Su, S. Boyd, and E. J. Candès, "A differential equation for modeling Nesterov's accelerated gradient method: Theory and insights," vol. 17, pp. 1–43, 2016.

Publication IV

Mihai I. Florea, Adrian Basarab, Denis Kouamé, and Sergiy A. Vorobyov. Restoration of Ultrasound Images using Spatially-variant Kernel Deconvolution. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, Calgary, Canada, pp. 796–800, Apr. 2018.

RESTORATION OF ULTRASOUND IMAGES USING SPATIALLY-VARIANT KERNEL DECONVOLUTION

Mihai I. Florea[⋆] *Adrian Basarab*[†] *Denis Kouamé*[†] *Sergiy A. Vorobyov*[⋆]

[⋆] Department of Signal Processing and Acoustics, Aalto University, Espoo, Finland
[†]IRIT UMR CNRS 5505, University of Toulouse, Toulouse, France

ABSTRACT

Most of the existing ultrasound image restoration methods consider a spatially-invariant point-spread function (PSF) model and circulant boundary conditions. While computationally efficient, this model is not realistic and severely limits the quality of reconstructed images. In this work, we address ultrasound image restoration under the hypothesis of piece-wise linear vertical variation of the PSF based on a small number of prototypes. No assumption is made on the structure of the prototype PSFs. To regularize the solution, we use the classical elastic net constraint. Existing methodologies are rendered impractical either due to their reliance on matrix inversion or due to their inability to exploit the strong convexity of the objective. Therefore, we propose an optimization algorithm based on the Accelerated Composite Gradient Method, adapted and optimized for this task. Our method is guaranteed to converge at a linear rate and is able to adaptively estimate unknown problem parameters. We support our theoretical results with simulation examples.

Index Terms— Accelerated Composite Gradient Method, point-spread function, reconstruction, restoration, spatially varying, ultrasound

1. INTRODUCTION

Ultrasound imaging is an efficient, cost effective, and safe medical imaging modality. It is widely used for various clinical applications and is especially well suited for the diagnosis of soft tissue pathologies. These advantages are however mitigated by the relative low image quality, in terms of signal-to-noise ratio, low contrast, and poor spatial resolution. The main factors affecting the quality of ultrasound images are the finite bandwidth and aperture of the imaging transducer as well as the physical phenomena (e.g., diffraction and attenuation) related to the propagation of sound waves in human tissues. Consequently, a rich body of scientific literature addresses ultrasound image reconstruction, i.e., the estimation of the tissue reflectivity function (TRF) from ultrasound images. Generally, existing approaches turn the TRF estimation into a deconvolution problem, by considering, under the first order Born approximation, that the formation of ultrasound images follows a 2D convolution model between the TRF and the system point-spread function (PSF). The PSF can be either estimated in a pre-processing step (see, e.g., [1–6]) or jointly estimated with the TRF, *i.e.*, blind deconvolution (see, e.g., [7–10]). Mainly for computational reasons, most of the existing ultrasound image restoration methods consider a spatially-invariant PSF model and circulant boundary conditions. However, independently of the acquisition setup, stationary convolution cannot accurately model the formation of ultrasound images. To overcome this issue, ultrasound images are generally divided in local regions prior to deconvolution, assuming a block-wise spatially-invariant PSF (see, e.g., [11]). To avoid issues related to stitching together the results of block-wise techniques, a few attempts have been very recently made in [12] and [13] to account for non-stationary convolution models in ultrasound imaging. The former relies on a very restrictive model with few degrees of freedom whereas the deconvolution method in the latter is computationally intractable for large images.

This paper proposes an optimization algorithm adapted to ultrasound image restoration under the hypothesis of spatially-variant PSF. The convolution kernel is assumed to be horizontally invariant but to vary vertically as a linear combination of neighboring prototype point-spread functions (PSF). To regularize the solution of the inverse problem generated by this model, we use herein the classical elastic net constraint [14]. Elastic net ensures a compromise between the ℓ_1-norm promoting sparse solutions and the ℓ_2-norm imposing smooth results. Its interest in ultrasound imaging has been already shown through different applications, e.g. compressed sensing [15], beamforming [16] or clutter filtering [17]. The elastic net regularized inverse problem has a non-differentiable objective and a large number of optimization variables. Therefore, it can only be addressed using proximal gradient schemes. The proposed method, based on the Accelerated Composite Gradient Method (ACGM) [18, 19], is able to simultaneously exploit the strong convexity of the problem and automatically provide a tight local estimate of the otherwise unknown Lipschitz constant.

2. PROBLEM FORMULATION

The acquisition model considered in this work is given by

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{P}\boldsymbol{x} + \boldsymbol{n}, \tag{1}$$

where $\boldsymbol{y}$ denotes the observed image, $\boldsymbol{x}$ is the tissue reflectivity function (TRF) to be recovered and $\boldsymbol{n}$ represents independent identically distributed additive white Gaussian noise. All images are of the same size, $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{n} \in \mathcal{D}_f \stackrel{\text{def}}{=} \mathbb{R}^{m_t \times n_t}$, where m_t denotes the height (number of pixels along the axial dimension) and n_t gives the width (lateral pixel count) of the TRF. The linear operator $\boldsymbol{P}$ pads the TRF with a $m_r \times n_r$ boundary, yielding an image of size $m_p \times n_p$, where $m_p = m_t + 2m_r$ and $n_p = n_t + 2n_r$, respectively. Our model accounts for any simple form of padding[1].

The linear operator $\boldsymbol{H}$ performs the spatially-variant kernel convolution. Within a classical pulse-echo acquisition scheme, focused

Part of this work has been supported by the Academy of Finland, under grant no. 299243, and CIMI Labex, Toulouse, France, under grant ANR-11-LABX-0040-CIMI within the program ANR-11-IDEX-0002-02.

[1]See [20] for a detailed discussion on simple padding and its implementation.

ultrasound waves are sequentially emitted by a sliding active subaperture. For each emission, the raw data is collected by the elements within the active aperture and further beamformed to compute an A-line representing one column of the ultrasound image. For this reason, the PSF can be reasonably considered spatially-invariant in the lateral (horizontal) direction. However, despite dynamic focusing in reception and time gain compensation, the PSF strongly varies in the axial (vertical) direction, *i.e.*, in the direction of ultrasound wave propagation, degrading spatial resolution away from the focal depth.

In our model, we account for this axial variation by assuming that a small number n_k of prototype kernels is known, each prototype PSF $\boldsymbol{K}_q$ having a center at row c_q for all $q \in \{1, ..., n_k\}$. The prototype PSFs are sorted by c_q and thus the values of c_q form a partition of the set of rows. The kernels of each row are computed using linear interpolation. Specifically, kernels above c_1 are equal to $\boldsymbol{K}_1$, those below c_{n_k} equal $\boldsymbol{K}_{n_k}$. The kernels of all other rows are obtained as a convex combination (alpha-blending) of the prototype PSF above and the one below that row, the proportion given by the relative distance to the two centers.

Every row produced by the linear operator $\boldsymbol{H}$ is obtained by taking the corresponding row in the input image (padded TRF) along with all pixels within a boundary of size $m_p \times n_p$ and performing discrete valid convolution with the kernel pertaining to that row, obtained as explained above. It follows that the padding boundary size has to match the prototype PSF radii. The use of discrete valid convolution is the reason behind the need for padding the TRF with $\boldsymbol{P}$.

Our acquisition model can be used to construct a deconvolution problem which seeks to minimize the additive white Gaussian noise subject to regularization. When employing the elastic net, the TRF can be obtained by solving the following optimization problem

$$\min_{\boldsymbol{x} \in \mathcal{D}_f} \frac{1}{2}\|\boldsymbol{HPx} - \boldsymbol{y}\|_2^2 + \lambda_1\|\boldsymbol{x}\|_1 + \frac{\lambda_2}{2}\|\boldsymbol{x}\|_2^2. \tag{2}$$

3. PROPOSED METHOD

To efficiently solve optimization problem (2) for any non-negative value of λ_2, we propose a variant of the Accelerated Composite Gradient Method (ACGM) [18, 19] optimized for the elastic net[2].

The objective F in problem (2) can be split into a quadratic function f and an elastic net regularizer Ψ as follows:

$$f(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{Ax} - \boldsymbol{y}\|_2^2, \quad \Psi(\boldsymbol{x}) = \lambda_1\|\boldsymbol{x}\|_1 + \frac{\lambda_2}{2}\|\boldsymbol{x}\|_2^2, \tag{3}$$

where $\boldsymbol{A} \stackrel{\text{def}}{=} \boldsymbol{HP}$. Function f is quadratic and consequently has Lipschitz continuous gradient. The Lipschitz constant L_f is given by $\sigma_{max}^2(\boldsymbol{A})$, where $\sigma_{max}(\boldsymbol{A})$ is the maximum eigenvalue of operator $\boldsymbol{A}$. In practice, $\sigma_{max}(\boldsymbol{A})$ may be intractable to compute and, as we shall see, L_f need not be known at all to ACGM. However it is known that operator $\boldsymbol{A}$ is ill-conditioned and we can assume that function f has strong convexity parameter $\mu_f = 0$. Elastic net regularizer Ψ is not differentiable due to the l_1 term but has strong convexity parameter $\mu_\Psi = \lambda_2$. Hence, the objective as a whole has a strong convexity parameter of $\mu = \mu_\Psi = \lambda_2$.

ACGM does not require the exact form of the objective function. It instead relies on calls to its zeroth and first order operations, collectively referred to as "oracle functions" [22]. The splitting of the reconstruction problem in (3) yields four such oracle functions: $f(\boldsymbol{x})$, $\Psi(\boldsymbol{x})$, $\nabla f(\boldsymbol{x})$, and $\text{prox}_{\tau\Psi}(\boldsymbol{x})$, for $\boldsymbol{x} \in \mathbb{R}^n$ and $\tau > 0$. The gradient is given by

$$\nabla f(\boldsymbol{x}) = \boldsymbol{A}^T(\boldsymbol{Ax} - \boldsymbol{y}), \tag{4}$$

where $\boldsymbol{A}^T = \boldsymbol{P}^T\boldsymbol{H}^T$. The proximal operator, defined as

$$\text{prox}_{\tau\Psi}(\boldsymbol{x}) \stackrel{\text{def}}{=} \underset{\boldsymbol{z} \in \mathbb{R}^{m_t \times n_t}}{\arg\min} \left(\Psi(\boldsymbol{z}) + \frac{1}{2\tau}\|\boldsymbol{z} - \boldsymbol{x}\|_2^2\right), \tag{5}$$

can be written in closed form (see also [19] and [23]) as

$$\text{prox}_{\tau\Psi}(\boldsymbol{x}) = \frac{1}{1 + \tau\lambda_2}\mathcal{T}_{\tau\lambda_1}(\boldsymbol{x}), \tag{6}$$

where the shrinkage operator $\mathcal{T}_\tau(\boldsymbol{x})$ is given by

$$\begin{gathered}(\mathcal{T}_\tau(\boldsymbol{x}))_{i,j} \stackrel{\text{def}}{=} (|\boldsymbol{x}_{i,j}| - \tau)_+ \operatorname{sgn}(\boldsymbol{x}_{i,j}), \\ \tau > 0, \quad i \in \{1, ..., m_t\}, \quad j \in \{1, ..., n_t\}.\end{gathered} \tag{7}$$

The structure of the objective function makes it possible to reduce the computational complexity of ACGM by departing from the oracle model. To estimate the local Lipschitz constant, operator $\boldsymbol{A}$ has to be applied at every iteration k to the new iterate $\boldsymbol{x}^{(k+1)}$. It is computationally inexpensive to cache these results by maintaining alongside the main iterate sequence $\boldsymbol{x}^{(k)}$, $k \in \{0, ..., k_{max}\}$ the sequence $\tilde{\boldsymbol{x}}^{(k)} = \boldsymbol{A}\tilde{\boldsymbol{x}}^{(k)}$. ACGM can be brought into an extrapolated form whereby an auxiliary point $\boldsymbol{z}^{(k+1)}$ can be obtained through linear extrapolation from $\boldsymbol{x}^{(k)}$ and $\boldsymbol{x}^{(k-1)}$. The new iterate $\boldsymbol{x}^{(k+1)}$ is computed based on the gradient of f at $\boldsymbol{z}^{(k+1)}$. The computational intensity of gradient expression (4) can be reduced as follows:

$$\nabla f(\boldsymbol{z}^{(k+1)}) = \boldsymbol{A}^T(\tilde{\boldsymbol{z}}^{(k+1)} - \boldsymbol{y}), \tag{8}$$

where

$$\tilde{\boldsymbol{z}}^{(k+1)} \stackrel{\text{def}}{=} \boldsymbol{Az}^{(k+1)} = \tilde{\boldsymbol{x}}^{(k)} + \beta(\tilde{\boldsymbol{x}}^{(k)} - \tilde{\boldsymbol{x}}^{(k-1)}), \tag{9}$$

and β is the extrapolation factor.

The line-search stopping criterion [19] of ACGM at every iteration k is given by

$$\begin{aligned}f(\boldsymbol{x}^{(k+1)}) \leq f(\boldsymbol{z}^{(k+1)}) + \nabla f(\boldsymbol{z}^{(k+1)})^T(\boldsymbol{x}^{(k+1)} - \boldsymbol{z}^{(k+1)}) \\ + \frac{L^{(k+1)}}{2}\|\boldsymbol{x}^{(k+1)} - \boldsymbol{z}^{(k+1)}\|_2^2,\end{aligned} \tag{10}$$

where $L^{(k+1)}$ is the Lipschitz constant estimate at iteration k. Substituting gradient expression (4) and rearranging terms yields

$$\|\boldsymbol{A}(\boldsymbol{x}^{(k+1)} - \boldsymbol{z}^{(k+1)})\|_2^2 \leq L^{(k+1)}\|\boldsymbol{x}^{(k+1)} - \boldsymbol{z}^{(k+1)}\|_2^2. \tag{11}$$

We obtain a computationally efficient expression by reusing the precomputed values $\tilde{\boldsymbol{x}}^{(k+1)}$ and $\tilde{\boldsymbol{z}}^{(k+1)}$ as

$$\|\tilde{\boldsymbol{x}}^{(k+1)} - \tilde{\boldsymbol{z}}^{(k+1)}\|_2^2 \leq L^{(k+1)}\|\boldsymbol{x}^{(k+1)} - \boldsymbol{z}^{(k+1)}\|_2^2. \tag{12}$$

The global Lipschitz constant L_f can alternatively be expressed as

$$L_f = \sup_{\mathcal{D}_f} \frac{\|\boldsymbol{Ax}\|_2^2}{\|\boldsymbol{x}\|_2^2}. \tag{13}$$

In practice, the estimates are below this value and we set the initial one to

$$L^{(0)} = \frac{\|\boldsymbol{Ax}_0\|_2^2}{\|\boldsymbol{x}_0\|_2^2} = \frac{\|\tilde{\boldsymbol{x}}_0\|_2^2}{\|\boldsymbol{x}_0\|_2^2}. \tag{14}$$

Incorporating the above performance enhancements into ACGM in extrapolated form yields the method listed in Algorithm 1.

[2] A simpler version of ACGM was introduced in [21] to deal with the case of $\lambda_2 = 0$. The proposed method can be regarded as a computationally efficient generalization of this earlier scheme.

Algorithm 1 Proposed method

Input: $\boldsymbol{x}_0$, λ_1, λ_2, k_{max}
$\tilde{\boldsymbol{x}}^{(0)} := \boldsymbol{H}\boldsymbol{P}\boldsymbol{x}^{(0)}$
$\boldsymbol{x}^{(-1)} = \boldsymbol{x}^{(0)}$
$\tilde{\boldsymbol{x}}^{(-1)} = \tilde{\boldsymbol{x}}_0$
$L^{(0)} = \|\tilde{\boldsymbol{x}}^{(0)}\|_2^2 / \|\boldsymbol{x}^{(0)}\|_2^2$
$q^{(0)} = \frac{\lambda_2}{L^{(0)}+\lambda_2}$
$t^{(0)} = 0$
for $k = 0, ..., k_{max} - 1$ **do**
  $\alpha := 1 - q^{(k)}(t^{(k)})^2$
  $L^{(k+1)} := r_d L^{(k)}$
  loop
    $q^{(k+1)} := \frac{\lambda_2}{L^{(k+1)}+\lambda_2}$
    $t^{(k+1)} := \frac{1}{2}\left(\alpha + \sqrt{\alpha^2 + 4\frac{L^{(k+1)}+\lambda_2}{L^{(k)}+\lambda_2}(t^{(k)})^2}\right)$
    $\beta := \frac{t^{(k)}-1}{t^{(k+1)}} \frac{1-q^{(k+1)}t^{(k+1)}}{1-q^{(k+1)}}$
    $\boldsymbol{z}^{(k+1)} := \boldsymbol{x}^{(k)} + \beta(\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(k-1)})$
    $\tilde{\boldsymbol{z}}^{(k+1)} := \tilde{\boldsymbol{x}}^{(k)} + \beta(\tilde{\boldsymbol{x}}^{(k)} - \tilde{\boldsymbol{x}}^{(k-1)})$
    $\tau := 1/L^{(k+1)}$
    $\boldsymbol{G} := \boldsymbol{P}^T\boldsymbol{H}^T(\tilde{\boldsymbol{z}}^{(k+1)} - \boldsymbol{y})$
    $\boldsymbol{x}^{(k+1)} := \frac{1}{1+\tau\lambda_2}\mathcal{T}_{\tau\lambda_1}(\boldsymbol{z}^{(k+1)} - \tau\boldsymbol{G})$
    $\tilde{\boldsymbol{x}}^{(k+1)} := \boldsymbol{H}\boldsymbol{P}\boldsymbol{x}^{(k+1)}$
    if $\|\tilde{\boldsymbol{x}}^{(k+1)} - \tilde{\boldsymbol{z}}^{(k+1)}\|_2^2 \leq L^{(k+1)}\|\boldsymbol{x}^{(k+1)} - \boldsymbol{z}^{(k+1)}\|_2^2$ **then**
      Break from loop
    else
      $L^{(k+1)} := r_u L^{(k+1)}$
    end if
  end loop
end for
Output: $\boldsymbol{x}^{(k_{max})}$

4. EXPERIMENTAL RESULTS

A three-step process was employed to simulate the RF ultrasound image: i) the calculation of the spatially-variant prototype PSFs; ii) the generation of the tissue reflectivity function (TRF); and iii) the spatially-variant convolution between the PSFs and the TRF, following the model described in (1) (Section 2).

The prototype PSFs were obtained in step i) using Field II, a realistic state-of-the art simulator [24, 25]. The simulation involved a linear 128-element ultrasound probe emitting ultrasound waves at a nominal frequency of 3 MHz. The width of each element was set to equal the wavelength (0.5 mm), while height was fixed at 5 mm. The distance between two consecutive elements was set to 0.1 mm. The transducer was excited using a two-period sinusoidal wave of frequency 3 MHz. The backscattered RF signals were sampled at a rate of 20 MHz. The prototype PSFs we obtained by placing isolated scatterers in front of the transducer with a distance in depth of 8.5 mm to each other. Ultrasound waves electronically focalized at 47 mm from the probe were emitted and the received echoes were statically focused prior to the delay-and-sum beamforming process. Hann apodization was used both for the emission and the reception.

The resulting $n_k = 10$ prototype PSFs are shown in Fig. 1. For the purpose of visualizing the areas influenced by individual prototype PSFs, they are displayed after envelope detection and min-max normalization centered at c_q for all $q \in \{1, ..., n_k\}$ in Fig. 1(a). To highlight the differences between individual prototype PSFs, they are displayed separately in Fig. 1(b). The 5th prototype PSF ($\boldsymbol{K}_5$) located at 43 mm from the probe was used in spatially-invariant deconvolution experiments. It is shown both in native form and after envelope detection in Fig. 2.

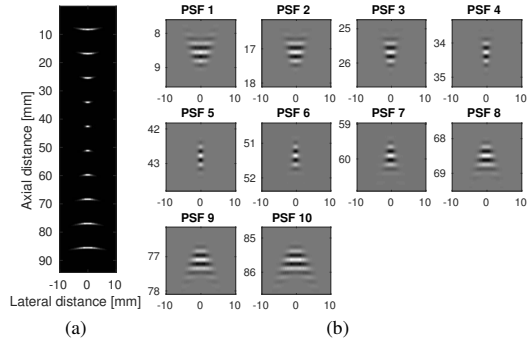


Fig. 1. Prototype PSFs generated with Field II for $n_k = 10$ depths at regularly spaced intervals of 8.5 mm. (a) Global view, after demodulation and min-max normalization, showing the location within the image of the prototype PSF centers; (b) Individual view, showing the spatial variance of the prototype PSFs.

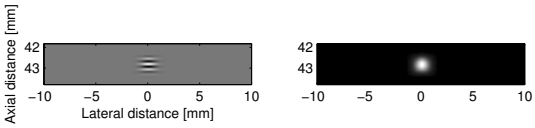


Fig. 2. The 5th prototype PSF $\boldsymbol{K}_5$ (left) and its demodulated version (right).

The TRF was obtained in step ii) following the classical procedure employed in ultrasound image simulation. A collection of uniform randomly located scatterers with zero-mean Gaussian random amplitudes has been generated. The standard deviation used to generate the amplitude of one scatterer depended on its location and was related, as suggested in the Field II simulator, to a digital image obtained from MRI and CT scans of a human kidney tissue. The number of scatterers was sufficiently large (10^5) to ensure fully developed speckle. The scatterer map was finally linearly interpolated onto a rectangular grid resulting into the TRF shown in Fig. 3(a).

In step iii), an ultrasound image was simulated from the TRF using the model in (1) to produce a starting point ($\boldsymbol{x}_0$) for the deconvolution experiments. First, the TRF was padded with a symmetric boundary. Next, the padded image was convolved with the spatially varying convolution operator $\boldsymbol{H}$, based on the prototype PSFs shown in Fig. 1. To simplify the hyperparameter tuning process, we have scaled $\boldsymbol{H}$ to ensure that $L^{(0)}$, as given by (14), is equal to 1. Finally, white Gaussian noise was added to the convolved image, such that the signal-to-noise ratio is 40 dB. The simulated ultrasound image is shown in B-mode representation in Fig. 3(b).

We have conducted two deconvolution experiments. Both used as starting point the simulated ultrasound image shown in Fig. 3(b) and the same values of the hyperparameters, $\lambda_1 = 0.005$ and $\lambda_2 = 0.01$, which were found by manual tuning to give the best results.

First, we have compared our method, which is able to integrate the spatial variability of the kernels in the deconvolution

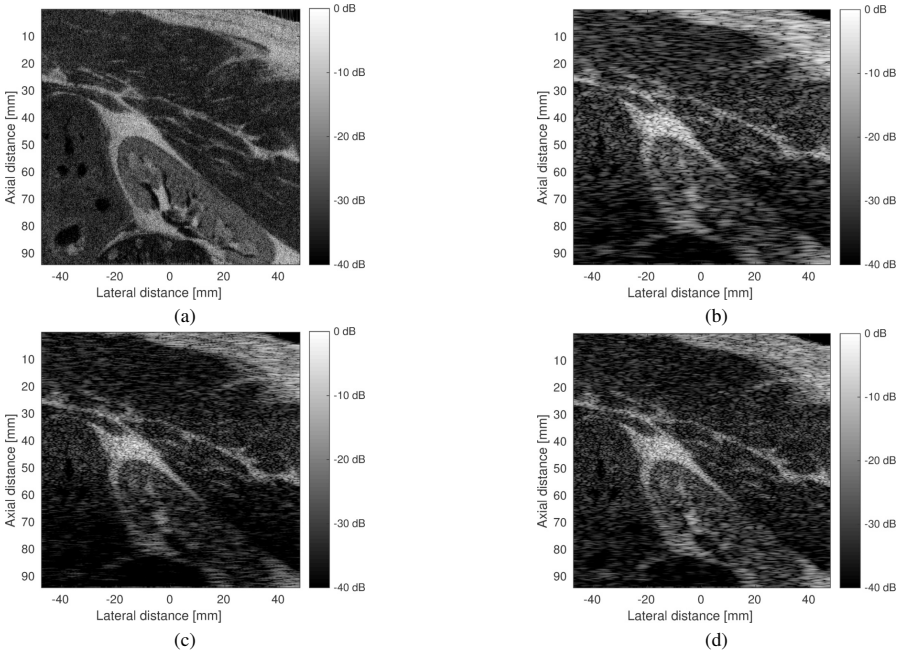


Fig. 3. (a) Ground truth of the tissue reflectivity function (TRF); (b) Observed B-mode image simulated from on the TRF in (a) using the image acquisition model in (1) employing the spatially-variant convolution operation based on the prototype PSFs in Fig. 1; (c) Spatially-invariant deconvolution result obtained using the fixed kernel $\boldsymbol{K}_5$ in Fig. 2; (d) Spatially-variant deconvolution result using our method.

process, with ACGM employing a spatially-invariant blur operator $\boldsymbol{H}$. Two restored images obtained after 150 iterations, are displayed in Fig. 3(c, d). The image in Fig. 3(c) was estimated considering a spatially-invariant PSF ($\boldsymbol{K}_5$ at 43 mm depth) and the one in Fig. 3(d) was obtained using our method. The quality of the deconvolution can be appreciated by comparing the restored images in Fig. 3(c, d) with the true TRF in Fig. 3(a). Note that the deconvolution results are also shown in B-mode. While the deconvolved images are similar in the vicinity of the focal point, our method manages to restore image features at the vertical extremities (Fig. 3(d)). These features are barely discernible both in the blurred image shown (Fig. 3(b)) as well as in the spatially-invariant PSF reconstruction (Fig. 3(c)). The simulation results support our previous claim that reconstruction quality of an image patch depends on the similarity between the blurring and the deblurring kernels applied to it, clearly demonstrating the superiority of our model.

The second experiment concerned optimization algorithms. We have compared the convergence behavior of our method to the state-of-the-art applicable to our model, in this case the Fast Iterative Shrinkage Thresholding Algorithm (FISTA) [26]. Fig. 4 shows the objective function values across iterations for the two methods. The convergence plot required an optimum value estimate $\hat{\boldsymbol{x}}^*$, which was obtained by running ACGM until floating point precision was exhausted. As predicted theoretically in [18, 19], the convergence rate of ACGM is linear. FISTA is unable to exploit the strong convexity of the objective and lags behind considerably.

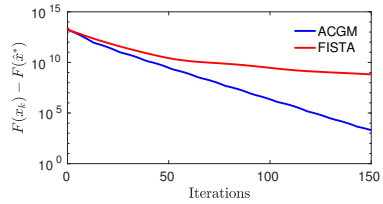


Fig. 4. Convergence rate of our method compared to FISTA.

5. CONCLUSION

We have devised a methodology for spatially-variant deconvolution of ultrasound images. Our method is theoretically guaranteed to converge linearly regardless of the structure of input data. In this respect, the reliability of our method is particularly of value to the stringent demands of the medical ultrasonography industry. Simulation results show that reconstruction is not only computationally tractable but has a rate that is competitive with existing approaches relying on far more restrictive assumptions. For that matter, our approach is able to address far more complex imaging models, even those that do not require horizontally-invariant PSF.

6. REFERENCES

[1] J. Ng, R. Prager, N. Kingsbury, G. Treece, and A. Gee, "Wavelet restoration of medical pulse-echo ultrasound images in an EM framework," *IEEE Trans. Ultrason. Ferroelectr. Freq. Control*, vol. 54, no. 3, pp. 550–568, 2007.

[2] R. Rangarajan, C. V. Krishnamurthy, and K. Balasubramaniam, "Ultrasonic imaging using a computed point spread function," *IEEE Trans. Ultrason. Ferroelectr. Freq. Control*, vol. 55, no. 2, pp. 451–464, Feb. 2008.

[3] H.-C. Shin, R. Prager, J. Ng, H. Gomersall, N. Kingsbury, G. Treece, and A. Gee, "Sensitivity to point-spread function parameters in medical ultrasound image deconvolution," *Ultrasonics*, vol. 49, no. 3, pp. 344 – 357, 2009.

[4] M. Alessandrini, S. Maggio, J. Poree, L. D. Marchi, N. Speciale, E. Franceschini, O. Bernard, and O. Basset, "A restoration framework for ultrasonic tissue characterization," *IEEE Trans. Ultrason. Ferroelectr. Freq. Control*, vol. 58, no. 11, pp. 2344–2360, 2011.

[5] C. Dalitz, R. Pohle-Frohlich, and T. Michalk, "Point spread functions and deconvolution of ultrasonic images," *IEEE Trans. Ultrason. Ferroelectr. Freq. Control*, vol. 62, no. 3, pp. 531–544, Mar. 2015.

[6] N. Zhao, A. Basarab, D. Kouamé, and J.-Y. Tourneret, "Joint segmentation and deconvolution of ultrasound images using a hierarchical Bayesian model based on generalized Gaussian priors," *IEEE Trans. Image Process.*, vol. 25, no. 8, pp. 3736–3750, 2016.

[7] O. Michailovich and D. Adam, "A novel approach to the 2-D blind deconvolution problem in medical ultrasound," *IEEE Trans. Med. Imag.*, vol. 24, pp. 86–104, Jan. 2005.

[8] O. Michailovich and A. Tannenbaum, "Blind deconvolution of medical ultrasound images: A parametric inverse filtering approach," *IEEE Trans. Image Process.*, vol. 16, no. 12, pp. 3005–3019, 2007.

[9] R. Jirik and T. Taxt, "Two dimensional blind Bayesian deconvolution of medical ultrasound images," *IEEE Trans. Ultrason. Ferroelectr. Freq. Control*, vol. 55, no. 10, pp. 2140–2153, 2008.

[10] C. Yu, C. Zhang, and L. Xie, "A blind deconvolution approach to ultrasound imaging," *IEEE Trans. Ultrason. Ferroelectr. Freq. Control*, vol. 59, no. 2, pp. 271–280, 2012.

[11] J. G. Nagy and D. P. O'Leary, "Restoring images degraded by spatially variant blur," *SIAM J. Sci. Comput.*, vol. 19, no. 4, pp. 1063–1082, 1998.

[12] O. V. Michailovich, "Non-stationary blind deconvolution of medical ultrasound scans," in *Proc. SPIE*, Mar. 2017, vol. 101391C.

[13] L. Roquette, M. M. J.-A. Simeoni, P. Hurley, and A. G. J. Besson, "On an Analytical, Spatially-Varying, Point-Spread-Function," in *Proc. IEEE Ultrason. Symp. (US)*, 2017.

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

[15] C. Quinsac, A. Basarab, and D. Kouamé, "Frequency domain compressive sampling for ultrasound imaging," *Advances in Acoustics and Vibration*, vol. 12, pp. 1–16, 2012.

[16] T. Szasz, A. Basarab, M.-F. Vaida, and D. Kouamé, "Elastic-net based beamforming in medical ultrasound imaging (regular paper)," in *Proc. IEEE Int. Symp. Biomed. Imaging (ISBI)*, Apr. 2016, Prague, Czech Republic, pp. 477–480.

[17] B. Byram, "Aperture domain model image reconstruction (admire) for improved ultrasound imaging," in *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Processing (ICASSP)*, Mar. 2017, pp. 6250–6253.

[18] M. I. Florea and S. A. Vorobyov, "An accelerated composite gradient method for large-scale composite objective problems," *arXiv preprint arXiv:1612.02352*, Dec. 2016.

[19] M. I. Florea and S. A. Vorobyov, "A generalized accelerated composite gradient method: Uniting Nesterov's Fast Gradient Method and FISTA," *arXiv preprint arXiv:1705.10266*, May 2017.

[20] M. I. Florea, A. Basarab, D. Kouamé, and S. A. Vorobyov, "An axially-variant kernel imaging model for ultrasound image reconstruction," *arXiv preprint arXiv:1801.08479*, Jan. 2018.

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

[22] A. Nemirovski, D. B. Yudin, and E. R. Dawson, *Problem complexity and method efficiency in optimization*, John Wiley & Sons, Inc., Hoboken, NJ, USA, 1983.

[23] N. Parikh, S. P. Boyd, et al., "Proximal algorithms," *Foundations and Trends in optimization*, vol. 1, no. 3, pp. 127–239, 2014.

[24] J. A. Jensen and N. B. Svendsen, "Calculation of pressure fields from arbitrarily shaped, apodized, and excited ultrasound transducers," *IEEE Trans. Ultrason. Ferroelectr. Freq. Control*, vol. 39, no. 2, pp. 262–267, Mar. 1992.

[25] J. A. Jensen, "Field: A program for simulating ultrasound systems," *Med. Biol. Eng. Comput.*, vol. 34, pp. 351–353, 1996.

[26] 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.

Publication V

Mihai I. Florea, Adrian Basarab, Denis Kouamé, and Sergiy A. Vorobyov. An Axially Variant Kernel Imaging Model Applied to Ultrasound Image Reconstruction. *IEEE Signal Processing Letters*, vol. 25, no.7, pp. 961–965, Jul. 2018.

An Axially Variant Kernel Imaging Model Applied to Ultrasound Image Reconstruction

Mihai I. Florea, *Student Member, IEEE*, Adrian Basarab, *Member, IEEE*, Denis Kouamé, *Member, IEEE*, and Sergiy A. Vorobyov, *Fellow, IEEE*

***Abstract*—Existing ultrasound deconvolution approaches unrealistically assume, primarily for computational reasons, that the convolution model relies on a spatially invariant kernel and circulant boundary conditions. We discard both restrictions and introduce an image formation model applicable to ultrasound imaging and deconvolution based on an axially varying kernel, which accounts for arbitrary boundary conditions. Our model has the same computational complexity as the one employing spatially invariant convolution and has negligible memory requirements. To accommodate the state-of-the-art deconvolution approaches when applied to a variety of inverse problem formulations, we also provide an equally efficient adjoint expression for our model. Simulation results confirm the tractability of our model for the deconvolution of large images. Moreover, in terms of accuracy metrics, the quality of reconstruction using our model is superior to that obtained using spatially invariant convolution.**

***Index Terms*—Axially varying, deconvolution, forward model, kernel, matrix-free, point-spread function, ultrasound.**

I. Introduction

ULTRASOUND imaging is a medical imaging modality widely adopted due to its efficiency, low cost, and safety. These advantages come at the expense of image quality. Consequently, the accurate estimation of the tissue reflectivity function (TRF) from ultrasound images is a subject of active research. Generally, the existing approaches assume that the formation of ultrasound images follows a two-dimensional (2-D) convolution model between the TRF and the system kernel. The convolution model is further constrained for computational reasons to have a spatially invariant kernel and circulant boundary conditions (see, e.g., [1]–[6]).

Pulse-echo emission of focused waves still remains the most widely used acquisition scheme in ultrasound imaging. It consists of sequentially transmitting narrow-focused beams. For each transmission centered at a lateral position, the raw data are used to beamform one radio frequency (RF) signal. Given the repeatability of the process in the lateral direction, the kernels do not vary laterally. However, despite dynamic focusing in reception and time gain compensation, the kernels become wider as we move away from the focal depth, thus, degrading the spatial resolution and motivating the proposed kernel variation model.

Previous works accounted for this variation by assuming kernel invariance over local regions and performing deconvolution blockwise (e.g., [7]). Very recently, ultrasound imaging convolution models with continuously varying kernels were proposed in [8] and [9]. However, the model presented in [8] makes the overly restrictive assumption that the spatially varying kernel is obtained from a constant reference kernel modulated by the exponential of a fixed discrete generator scaled by the varying kernel center image coordinates. Therefore, it does not take into account the depth-dependent spatial-resolution degradation explained previously. On the other hand, the deconvolution method proposed in [9] has an iteration complexity proportional to the cube of the number of pixels in the image, limiting its applicability to very small images.

The contributions of this letter are as follows.

1) We propose a novel axially variant kernel ultrasound image formation model (see Section III).
2) Our model is linear and may be implemented as a matrix. However, the matrix form does not scale because its complexity is proportional to the square of the number of pixels in the image. Therefore, we provide an efficient matrix-free implementation of axially varying convolution that entails the same computational cost as spatially invariant convolution (see Section III-B).
3) The deconvolution problem is ill-posed and many deconvolution models can only be solved approximately using proximal-splitting methods (see [10] and [11] and references therein) that compute the gradient of a data-fidelity term at every iteration. The data-fidelity gradient expression includes calls to both the model operator and its adjoint. We express this adjoint operator in a form of equal complexity to that of the forward model operator (see Section IV).
4) We confirm using simulation results that deconvolution with our model is tractable even for large images and produces results superior to those obtained by using the spatially invariant model (see Section V).

II. Notation

In this letter, images (ultrasound images and TRFs) are vectorized in column-major order but referenced in 2-D form. For

Manuscript received January 31, 2018; revised March 26, 2018; accepted March 26, 2018. Date of publication April 9, 2018; date of current version May 23, 2018. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. James E. Fowler. This work was supported in part by the Academy of Finland under Grant 299243 and in part by the CIMI Labex, Toulouse, France, under Grant ANR-11-LABX-0040-CIMI within the program ANR-11-IDEX-0002-02. *(Corresponding author: Mihai I. Florea.)*

M. I. Florea and S. A. Vorobyov are with the Department of Signal Processing and Acoustics, Aalto University, Aalto FI-00076, Finland (e-mail: mihai.florea@aalto.fi; sergiy.vorobyov@aalto.fi).

A. Basarab and D. Kouamé are with the IRIT, CNRS UMR 5505, University of Toulouse, Toulouse 31062, France (e-mail: basarab@irit.fr; kouame@irit.fr).

Digital Object Identifier 10.1109/LSP.2018.2824764

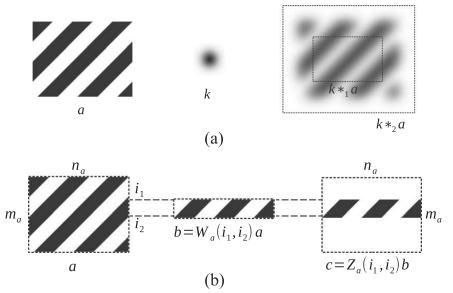


Fig. 1. (a) Convolving test image $\boldsymbol{a}$ with a Gaussian kernel $\boldsymbol{k}$. The inner rectangle represents valid convolution, whereas the outer one marks full convolution. (b) Applying the full-width window operator, followed by a full-width zero-padding operator on test image $\boldsymbol{a}$. Here, black and white correspond to values of 1 and 0, respectively. Kernel $\boldsymbol{k}$ is displayed after min–max normalization.

instance, image $\boldsymbol{v} \in \mathbb{R}^{m_v n_v}$ corresponds to an $m_v \times n_v$ 2-D image and has the pixel value at coordinates (i, j) given by $v_{m_v(j-1)+i}$. However, for clarity of exposition, we denote it as a 2-D object $\boldsymbol{v} \in \mathbb{R}^{m_v \times n_v}$, with the pixel value at location (i, j) given by $\boldsymbol{v}_{i,j}$. Bold marks this artificial indexing. Similarly, linear operators are matrices but referred to as 4-D tensors, e.g., $\boldsymbol{O} : \mathbb{R}^{m_v \times n_v} \to \mathbb{R}^{m_w \times n_w}$ denotes $\boldsymbol{O} \in \mathbb{R}^{m_v n_v \times m_w n_w}$.

In the sequel, we define several classes of linear operators that constitute the mathematical building blocks of our proposed model and its analysis. Note that these are more general than normal linear operators because their dimensions not only depend on those of their parameters but also on the dimensions of their arguments.

A. Convolution Operators

For all $m_k, n_k \geq 1$, all kernels $\boldsymbol{k} \in \mathbb{R}^{m_k \times n_k}$, and all $m_a \geq m_k$, $n_a \geq n_k$, we define the linear operators $\boldsymbol{\mathcal{C}}_1(\boldsymbol{k})$ and $\boldsymbol{\mathcal{C}}_2(\boldsymbol{k})$ as

$$\boldsymbol{\mathcal{C}}_1(\boldsymbol{k})\boldsymbol{a} \stackrel{\text{def}}{=} \boldsymbol{k} *_1 \boldsymbol{a}, \quad \boldsymbol{\mathcal{C}}_2(\boldsymbol{k})\boldsymbol{a} \stackrel{\text{def}}{=} \boldsymbol{k} *_2 \boldsymbol{a}$$

for all $\boldsymbol{a} \in \mathbb{R}^{m_a \times n_a}$, where operations $*_1$ and $*_2$ denote (discrete) valid convolution and full convolution, respectively, defined as

$$(\boldsymbol{k} *_1 \boldsymbol{a})_{i,j} \stackrel{\text{def}}{=} \sum_{p=1}^{m_k} \sum_{q=1}^{n_k} \boldsymbol{k}_{p,q} \boldsymbol{a}_{i-p+m_k, j-q+n_k}$$
$$i \in \{1, \ldots, m_a - m_k + 1\}, \quad j \in \{1, \ldots, n_a - n_k + 1\}$$

$$(\boldsymbol{k} *_2 \boldsymbol{a})_{i,j} \stackrel{\text{def}}{=} \sum_{p=\bar{p}_i}^{\bar{p}_i} \sum_{q=\bar{q}_j}^{\bar{q}_j} \boldsymbol{k}_{p,q} \boldsymbol{a}_{i-p+1, j-q+1}$$
$$i \in \{1, \ldots, m_a + m_k - 1\}, \quad j \in \{1, \ldots, n_a + n_k - 1\}$$
$$p_i = \max\{1, i - m_a + 1\}, \quad \bar{p}_i = \min\{i, m_k\}$$
$$q_j = \max\{1, j - n_a + 1\}, \quad \bar{q}_j = \min\{j, n_k\}.$$

The difference between the two forms of convolution is exemplified in Fig. 1(a). Valid convolution is thereby the subset of full convolution where every output pixel is expressed using the entire kernel $\boldsymbol{k}$.

B. Auxiliary Operators

For conciseness, we also introduce the following auxiliary operators. None involve any computation in practice.

Let the rotation operator $\boldsymbol{\mathcal{R}}(\boldsymbol{k})$ be given by

$$(\boldsymbol{\mathcal{R}}(\boldsymbol{k}))_{i,j} \stackrel{\text{def}}{=} \boldsymbol{k}_{m_k - i + 1, n_k - j + 1}$$
$$i \in \{1, \ldots, m_k\}, \quad j \in \{1, \ldots, n_k\}.$$

To further simplify the notation, we denote the exception index set $\mathcal{I}(a, b, c) \stackrel{\text{def}}{=} \{1, \ldots, c\} \backslash \{a, \ldots, b\}$ for all $1 \leq a \leq b \leq c$. The full-width window and zero-padding operators are defined as

$$(\boldsymbol{\mathcal{W}}_s(i_1, i_2)\boldsymbol{a})_{i,j} \stackrel{\text{def}}{=} \boldsymbol{a}_{i+i_1, j}, \quad i \in \{0, \ldots, i_2 - i_1\}$$

$$(\boldsymbol{\mathcal{Z}}_s(i_1, i_2)\boldsymbol{a})_{i,j} \stackrel{\text{def}}{=} \begin{cases} \boldsymbol{a}_{i-i_1, j}, & i \in \{i_1, \ldots, i_2\} \\ 0, & i \in \mathcal{I}(i_1, i_2, m_s) \end{cases}$$

where $j \in \{1, \ldots, n_s\}$ and index $s \in \{t, p\}$ stands for image size quantities m_t, m_p, n_t, and n_p. Their effect on a test image is shown in Fig. 1(b).

III. Axially Variant Kernel Based Ultrasound Imaging Model

We propose the following image formation model:

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{P}\boldsymbol{x} + \boldsymbol{n} \tag{1}$$

where $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{n} \in \mathbb{R}^{m_t \times n_t}$ denote the TRF to be recovered, the observed RF image, and the independent identically distributed additive white Gaussian noise (AWGN), respectively.

A. Padding

Operator $\boldsymbol{P} : \mathbb{R}^{m_t \times n_t} \to \mathbb{R}^{m_p \times n_p}$ pads the TRF with a boundary of width n_r and height m_r, yielding an image of size $m_p = m_t + 2m_r$ times $n_p = n_t + 2n_r$. Padding in our ultrasound imaging model allows us to reconstruct a TRF of the same size as the observed RF image. To this end, we must simulate the effects that the surrounding tissues have on the imaged tissues. Padding is an estimation of the surrounding tissues using information from the imaged TRF. This estimation only affects the border of the reconstructed TRF. If this border information is not required, the reconstructed TRF can simply be cropped accordingly. The addition of padding to our model brings the advantage of accommodating both options.

For computational reasons, $\boldsymbol{P}$ is assumed linear and separable along the dimensions of the image. Separability translates to $\boldsymbol{P} = \boldsymbol{P}_m \boldsymbol{P}_n$. Here, $\boldsymbol{P}_m$ pads every column of the image independently by applying the 1-D padding (linear) operator $\boldsymbol{\mathcal{P}}(m_t, m_r)$. Consequently, when $n_t = 1$ and $n_r = 0$, operators $\boldsymbol{P}$ and $\boldsymbol{\mathcal{P}}(m_t, m_r)$ are equivalent. The row component $\boldsymbol{P}_n$ treats every row as a column vector, applies $\boldsymbol{\mathcal{P}}(n_t, n_r)$ to it, and turns the result back into a row.

Padding, either in 1-D or 2-D, can be performed without explicitly deriving an operator matrix. However, the matrix form facilitates the formulation of the corresponding adjoint operator. Common matrix forms of operator $\boldsymbol{\mathcal{P}}(m_t, n_t)$ are shown in Fig. 2 for $m_t = 10$ and $m_r = 3$. These examples demonstrate that the matrix form of $\boldsymbol{\mathcal{P}}(m_t, m_r)$ can be easily generated programmatically and, due to its sparsity, can be stored in memory

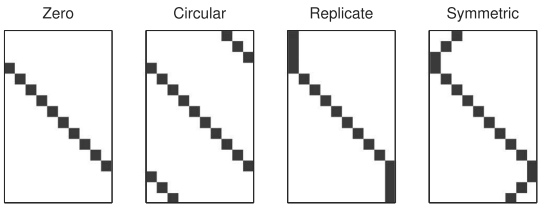


Fig. 2. Common matrix forms of 1-D padding operator $\boldsymbol{\mathcal{P}}(10,3)$. Black denotes a value of 1 and white denotes 0.

even for very large values of m_t and m_r. These properties extend to the matrix form of the 2-D padding operator $\boldsymbol{P}$ by virtue of the following result.

Theorem 1: Padding operator $\boldsymbol{P}$ can be obtained programatically in the form of a sparse matrix as

$$\boldsymbol{P} = \boldsymbol{\mathcal{P}}(n_t, n_r) \otimes \boldsymbol{\mathcal{P}}(m_t, m_r)$$

where $\otimes$ denotes the Kronecker product.

Proof. See [12, Appendix A]. ■

B. Axially Varying Convolution

Linear operator $\boldsymbol{H}$: $\mathbb{R}^{m_p \times n_p} \rightarrow \mathbb{R}^{m_t \times n_t}$ performs the axially variant kernel convolution. We define it as the operation whereby each *row* $i_h \in \{1, \ldots, m_t\}$ of the output image is obtained by the valid convolution between the kernel pertaining to that row $\boldsymbol{k}(i_h) \in \mathbb{R}^{m_k \times n_k}$, where $m_k = 2m_r + 1$ and $n_k = 2n_r + 1$, and the corresponding patch in the input (padded) TRF. The auxiliary operators defined in Section II enable us to write $\boldsymbol{H}$ as a sum of linear operators based on the observation that the concatenation of the output rows has the same effect as the summation of the rows appropriately padded with zeros. Analytically, this translates to

$$\boldsymbol{H} = \sum_{i_h=1}^{m_t} \boldsymbol{\mathcal{Z}}_t(i_h, i_h) \boldsymbol{\mathcal{C}}_1(\boldsymbol{k}(i_h)) \boldsymbol{\mathcal{W}}_p(i_h, i_h + 2m_r). \quad (2)$$

In matrix form, operator $\boldsymbol{H}$ would need to store $m_p n_p m_t n_t$ coefficients and its invocation would entail an equal number of multiplications. Its complexity would, thus, be greater than the square of the number of pixels in the image, limiting its applicability to medium-sized images. Using the matrix-free expression in (2), operator $\boldsymbol{H}$ performs $m_k n_k m_t n_t$ multiplications and has negligible memory requirements. Therefore, in ultrasound imaging, the matrix-free representation is not only vastly superior to its matrix counterpart (because the kernel is much smaller than the image), but also has the same computational complexity as the spatially invariant convolution operation (excluding the unrealistic circulant boundary case).

Unlike the forward model which, by utilizing operators $\boldsymbol{H}$ and $\boldsymbol{P}$, can be computed exactly with great efficiency, many deconvolution models can only be solved approximately using proximal-splitting methods that optimize an objective containing a data-fidelity term $\phi(\boldsymbol{HPx} - \boldsymbol{y})$. These methods employ at every iteration the gradient of the data-fidelity term, given by

$$\nabla(\phi(\boldsymbol{HPx} - \boldsymbol{y})) = \boldsymbol{P}^T \boldsymbol{H}^T (\nabla\phi)(\boldsymbol{HPx} - \boldsymbol{y}). \quad (3)$$

Note that, under our AWGN assumption, ϕ is the square of the ℓ_2-norm but the results in this letter may be applied to other additive noise models.

The gradient expression in (3) depends on $\boldsymbol{H}$ and $\boldsymbol{P}$ as well as their adjoints. In the following, we derive computationally efficient expressions for adjoint operators $\boldsymbol{H}^T$ and $\boldsymbol{P}^T$.

IV. Adjoint of Model Operator

By taking the adjoint in (2), we get

$$\boldsymbol{H}^T = \sum_{i_h=1}^{m_t} (\boldsymbol{\mathcal{W}}_p(i_h, i_h + 2m_r))^T (\boldsymbol{\mathcal{C}}_1(\boldsymbol{k}(i_h)))^T (\boldsymbol{\mathcal{Z}}_t(i_h, i_h))^T.$$

To obtain a matrix-free representation of $\boldsymbol{H}^T$, we need the corresponding matrix-free expressions for the adjoints of the convolution and auxiliary operators. First, it trivially holds that the window operator and the corresponding zero-padding operator are mutually adjoint expressed as

$$(\boldsymbol{\mathcal{W}}_s(i_1, i_2))^T = \boldsymbol{\mathcal{Z}}_s(i_1, i_2). \quad (4)$$

The adjoint of valid convolution can be linked to full convolution as follows.

Theorem 2: The adjoint of valid convolution is full correlation (convolution with the rotated kernel), namely

$$(\boldsymbol{\mathcal{C}}_1(\boldsymbol{k}))^T = \boldsymbol{\mathcal{C}}_2(\boldsymbol{\mathcal{R}}(\boldsymbol{k})).$$

Proof. See [12, Appendix B]. ■

Theorem 2 and (4) yield a matrix-free expression for $\boldsymbol{H}^T$ in the form of

$$\boldsymbol{H}^T = \sum_{i_h=1}^{m_t} \boldsymbol{\mathcal{Z}}_p(i_h, i_h + 2m_r) \boldsymbol{\mathcal{C}}_2(\boldsymbol{\mathcal{R}}(\boldsymbol{k}(i_h))) \boldsymbol{\mathcal{W}}_t(i_h, i_h). \quad (5)$$

Therefore, operators $\boldsymbol{H}$ and $\boldsymbol{H}^T$ have equal computational complexity. Moreover, they exhibit two levels of parallelism. The convolution operators themselves are fully parallel and the computations pertaining to each row i_h can be performed concurrently. Thus, in matrix-free from, both operators benefit from parallelization in the same way as their matrix counterparts.

The adjoint of the padding operator $\boldsymbol{P}^T$ can be obtained either directly through sparse matrix transposition or by applying transposition in Theorem 1 as

$$\boldsymbol{P}^T = (\boldsymbol{P})^T = (\boldsymbol{\mathcal{P}}(n_t, n_r))^T \otimes (\boldsymbol{\mathcal{P}}(m_t, m_r))^T. \quad (6)$$

Finally, note that whereas the column-major order assumption can be made without loss of generality for operator $\boldsymbol{H}$, it is not the case for the padding operator $\boldsymbol{P}$. In particular, the row-major vectorization assumption reverses the terms in the Kronecker product.

V. Experimental Results

We have tested our model on a simulated ultrasound image deconvolution problem. The ground truth TRF, as shown in Fig. 3(a), was computed by interpolating to a grid Gaussian distributed random scatterers with standard deviations (SDs) determined by a pixel intensity map (see the kidney phantom from the Field II simulator [13], [14]). The map is a patch from an optical scan of human kidney tissue. The TRF is $m_t = 2480$ by $n_t = 480$ pixels in size, corresponding to 94 mm $\times$ 95 mm.

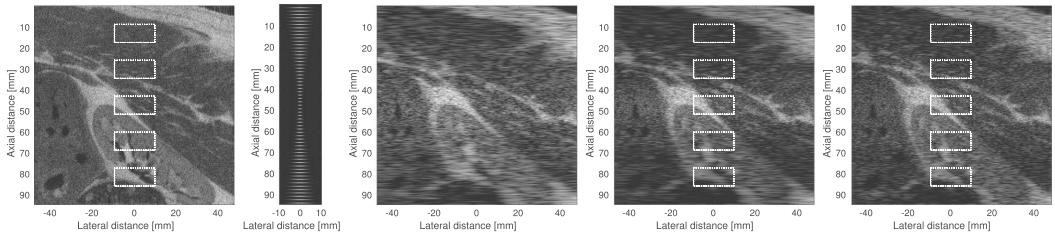


Fig. 3. (a) Ground truth (in B-mode) of the TRF. (b) Demodulated kernels $\boldsymbol{k}(i_h)$ for 20 depths at regularly spaced intervals of 2 mm. (c) Observed B-mode image simulated following the proposed axially variant convolution model. (d) AI deconvolution result, IR in B-mode, obtained with a fixed kernel equal to $\boldsymbol{k}(m_t/2)$ (the center kernel of the AV model). (e) AV deconvolution result, VR in B-mode, using our model. All the images are displayed using a dynamic range of 40 dB. White rectangles mark the patches used in computing the quality metrics.

More TRFs and their corresponding simulation results can be found in [12].

For every row $i_h \in \{1, \ldots, m_t\}$, we have defined the kernel $\boldsymbol{k}(i_h)$ in (2) as

$$\boldsymbol{k}(i_h)_{i,j} = \rho_{\mu_z,\sigma_z}(i)\rho_{\mu_x,\sigma_x(i_h)}(j)\cos\left(2\pi f_0/f_s(i-\mu_z)\right)$$

where $\rho_{\mu,\sigma}(x)$ is a normalized Gaussian window, given by $\rho_{\mu,\sigma}(x) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$, and parameters μ_z and μ_x are the center coordinates of the kernel. Axial SD was set to $\sigma_z = \sigma_1$ and lateral SD to $\sigma_x(i_h) = \sqrt{((2i_h)/m_t - 1)^2(\sigma_2^2 - \sigma_1^2) + \sigma_1^2}$, with $\sigma_1 = m_r/3$ and $\sigma_2 = n_r/3$. Here, $f_0 = 3$ MHz and $f_s = 20$ MHz are the ultrasound central and sampling frequencies, respectively. The depth-dependent width variation of the kernel simulates the lateral spatial-resolution degradation when moving away from the focus point, located in this experiment at the center of the image (47 mm from the probe). The envelopes of these kernels at regular intervals across the image are shown in Fig. 3(b). We chose symmetric padding, as illustrated in Fig. 2, because it is more realistic than circular padding and zero padding and, by using a larger number of pixels from the TRF, more robust to noise than replicate padding. A small amount of noise was added such that the signal-to-noise ratio is 40 dB. The ultrasound image produced from the TRF using our forward model in (1) is shown in Fig. 3(c).

To estimate the TRF, we have considered an elastic net [15] regularized least squares (based on the AWGN assumption) deconvolution model

$$\min_{\boldsymbol{x}} \frac{1}{2}\|\boldsymbol{HPx} - \boldsymbol{y}\|_2^2 + \lambda_1\|\boldsymbol{x}\|_1 + \frac{\lambda_2}{2}\|\boldsymbol{x}\|_2^2$$

with manually tuned parameters $\lambda_1 = 2e-3$ and $\lambda_2 = 1e-4$. For deconvolution, we have employed the accelerated composite gradient method (ACGM) [16], [17] on account of its low resource usage, applicability, adaptability, and near-optimal linear-convergence rate on elastic net regularized optimization problems.

Every iteration of ACGM is dominated by the computationally intensive data-fidelity gradient function $\nabla f(\boldsymbol{x}) = \boldsymbol{P}^T\boldsymbol{H}^T(\boldsymbol{HPx} - \boldsymbol{y})$. All other calculations performed by ACGM are either negligible when compared to $\nabla f(\boldsymbol{x})$ or can be reduced to subexpressions of $\nabla f(\boldsymbol{x})$.

Due to the efficient matrix-free expressions of $\boldsymbol{H}$ in (2) and $\boldsymbol{H}^T$ in (5) as well as the sparse matrix implementation of $\boldsymbol{P}$ and $\boldsymbol{P}^T$ (easily precomputed using Theorem 1 and (6), respectively), deconvolution with our model entails the same computational cost as with a fixed-kernel model.

TABLE I
ACCURACY METRICS COMPUTED FOR FIVE PATCHES IN THE RECONSTRUCTED IMAGES IR AND VR

Patch center row	IR NRMSE	IR MSSIM	VR NRMSE	VR MSSIM
13 mm	0.0972	99.45%	0.0291	99.84%
30 mm	0.0978	99.43%	0.0317	99.81%
47 mm	0.1571	96.55%	0.1194	97.01%
64 mm	0.1088	98.74%	0.0635	99.06%
81 mm	0.1267	97.72%	0.0888	98.05%

The result of axially invariant (AI) deconvolution, IR, is shown in Fig. 3(d), and using our axially variant (AV) model, VR, in Fig. 3(e), both after 150 iterations. The normalized root-mean-square error (NRMSE) and the mean image structural similarity (MSSIM) [18] accuracy metrics were computed for five patches in IR and VR after Gaussian normalization and envelope detection. The values are listed in Table I.

Our approach achieves almost perfect low-frequency reconstruction across the TRF. The gain in reconstruction quality is evident, especially in the upper and lower extremities, as can be discerned both empirically from Fig. 3 as well as from the accuracy metric discrepancy in the corresponding patches (see Table I), particularly the NRMSE. Interestingly, even though the two models differ only slightly at the center of the image, our model performs better in that region as well.

VI. CONCLUSION

In this letter, we have proposed an axially varying convolution forward model for ultrasound imaging. The physics of ultrasound image formation as well as our deconvolution simulation results show the superiority of our model over the traditional fixed-kernel model.

Our matrix-free formulae for the adjoints of the convolution and auxiliary operators, necessary for the implementation of deconvolution using proximal-splitting techniques, also constitute a solid theoretical foundation for deconvolution methodologies using more sophisticated models, particularly those where the kernel also varies along the lateral direction. Furthermore, our theoretical results and methodology are not restricted to ultrasound imaging only and may be extrapolated to other imaging modalities and applications as well.

REFERENCES

[1] J. Ng, R. Prager, N. Kingsbury, G. Treece, and A. Gee, "Wavelet restoration of medical pulse-echo ultrasound images in an EM framework," *IEEE Trans. Ultrason., Ferroelect., Freq. Control*, vol. 54, no. 3, pp. 550–568, Mar. 2007.

[2] R. Rangarajan, C. V. Krishnamurthy, and K. Balasubramaniam, "Ultrasonic imaging using a computed point spread function," *IEEE Trans. Ultrason., Ferroelect., Freq. Control*, vol. 55, no. 2, pp. 451–464, Feb. 2008.

[3] H.-C. Shin *et al.*, "Sensitivity to point-spread function parameters in medical ultrasound image deconvolution," *Ultrasonics*, vol. 49, no. 3, pp. 344–357, 2009.

[4] M. Alessandrini *et al.*, "A restoration framework for ultrasonic tissue characterization," *IEEE Trans. Ultrason., Ferroelect., Freq. Control*, vol. 58, no. 11, pp. 2344–2360, Nov. 2011.

[5] C. Dalitz, R. Pohle-Frohlich, and T. Michalk, "Point spread functions and deconvolution of ultrasonic images," *IEEE Trans. Ultrason., Ferroelect., Freq. Control*, vol. 62, no. 3, pp. 531–544, Mar. 2015.

[6] N. Zhao, A. Basarab, D. Kouamé, and J.-Y. Tourneret, "Joint segmentation and deconvolution of ultrasound images using a hierarchical Bayesian model based on generalized Gaussian priors," *IEEE Trans. Image Process.*, vol. 25, no. 8, pp. 3736–3750, Aug. 2016.

[7] J. G. Nagy and D. P. O'Leary, "Restoring images degraded by spatially variant blur," *SIAM J. Sci. Comput.*, vol. 19, no. 4, pp. 1063–1082, Jul. 1998.

[8] O. V. Michailovich, "Non-stationary blind deconvolution of medical ultrasound scans," in *Proc. Int. Soc. Opt. Eng.*, Bellingham, WA, USA, Mar. 2017, Paper 101391C.

[9] L. Roquette, M. M. J.-A. Simeoni, P. Hurley, and A. G. J. Besson, "On an analytical, spatially-varying, point-spread-function," in *Proc. 2017 IEEE Int. Ultrasound Symp.*, Sep. 2017, Washington, DC, USA, pp. 1–4.

[10] P. L. Combettes and J.-C. Pesquet, "Proximal splitting methods in signal processing," in *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*. New York, NY, USA: Springer-Verlag, 2011, pp. 185–212.

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

[12] M. I. Florea, A. Basarab, D. Kouamé, and S. A. Vorobyov, "An axially-variant kernel imaging model for ultrasound image reconstruction," *arXiv:1801.08479*, 2018.

[13] J. A. Jensen, "Field: A program for simulating ultrasound systems," *Med. Biol. Eng. Comput.*, vol. 34, pp. 351–353, 1996.

[14] J. A. Jensen and N. B. Svendsen, "Calculation of pressure fields from arbitrarily shaped, apodized, and excited ultrasound transducers," *IEEE Trans. Ultrason., Ferroelect., Freq. Control*, vol. 39, no. 2, pp. 262–267, Mar. 1992.

[15] 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.

[16] M. I. Florea and S. A. Vorobyov, "An accelerated composite gradient method for large-scale composite objective problems," arXiv:1612.02352, Apr. 16, 2018.

[17] M. I. Florea and S. A. Vorobyov, "A generalized accelerated composite gradient method: Uniting Nesterov's fast gradient method and FISTA," arXiv:1705.10266, Apr. 16, 2018.

[18] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

In the last years, we can see an increasing interest to the algorithms for solving large-scale optimization problem. [...] At the same time, the modern style of justification of the numerical methods assumes a detailed complexity analysis of their worst-case behavior. This can help in identification of their natural mode of implementation (online versus offline). [The proposed thesis contains the] development of new Accelerated Gradient Methods for minimizing a convex function in Composite Form. The corresponding complexity analysis is based on a variant of the estimating sequences technique. Moreover, the proposed schemes do not need an a priori knowledge of the Lipschitz constant for the gradient of the smooth part of the objective function. They can be used as for minimizing strongly and non-strongly convex function. Some of them can ensure a monotone decrease of the value of the objective function. [...] I believe that this is an excellent thesis, which contains very interesting and original results.

Professor Yurii Nesterov

Aalto-DD 197/2018

ISBN 978-952-60-8226-4 (printed)
ISBN 978-952-60-8227-1 (pdf)
ISSN 1799-4934 (printed)
ISSN 1799-4942 (pdf)

Aalto University
School of Electrical Engineering
Department of Signal Processing and Acoustics
www.aalto.fi