ISE

Industrial and
Systems Engineering

Gradient Flows and Accelerated Proximal Splitting Methods

Guilherme França[1], Daniel P. Robinson[2], and René Vidal[1]

[1]Mathematical Institute for Data Science, Johns Hopkins University

[2]Department of Industrial and Systems Engineering, Lehigh University

ISE Technical Report 19T-020

LEHIGH
UNIVERSITY

Gradient Flows and Accelerated Proximal Splitting Methods

Guilherme França,[a*] Daniel P. Robinson,[b] and René Vidal[a]

[a] Mathematical Institute for Data Science, Johns Hopkins University
[b] Department of Industrial and Systems Engineering, Lehigh University

Abstract

Proximal algorithms are well-suited for nonsmooth and constrained large-scale optimization problems and therefore suitable for applications in many areas of science. There are essentially four proximal algorithms based on fixed-point iterations currently known: forward-backward splitting, forward-backward-forward or Tseng splitting, Douglas-Rachford, and the Davis-Yin three operator splitting. In addition, the alternating direction method of multipliers (ADMM) is also closely related. In this paper we show that all of these algorithms can be derived as different discretizations of a single differential equation, namely the simple gradient flow. This is achieved through splitting methods for differential equations. Moreover, employing similar discretization schemes to a particular second-order differential equation, which we refer to as the accelerated gradient flow, results in accelerated variants of each respective proximal algorithm; we simultaneously consider two types of acceleration, although other choices are also possible. For instance, we propose accelerated variants of Davis-Yin and Tseng splitting, as well as accelerated extensions of ADMM. Interestingly, we show that ADMM and its accelerated variants correspond to rebalanced splittings, which is a recent technique designed to preserve steady states of the underlying differential equation. We show that all derived algorithms are valid first-order integrators under suitable assumptions. Our results strengthen the connections between optimization and continuous dynamical systems, offer a unified perspective on accelerated algorithms, and provide new accelerated algorithms.

[*] guifranca@gmail.com

Contents

1 Introduction

Acceleration strategies in the context of optimization have proved to be powerful. One example, termed *heavy ball acceleration*, was introduced by Polyak [1], while perhaps the most influential form of acceleration was introduction by Nesterov [2] and is often called *Nesterov acceleration*. Although neither are intuitive in their precise design, it has recently been shown that both can be obtained as explicit Euler discretizations of a certain second-order ordinary differential equation (ODE), and correspond to accelerated variants of gradient descent. This perspective has helped demystify the "magic" of acceleration techniques in optimization.

There is an increasing interest in understanding the connections between optimization and continuous dynamical systems, especially for accelerated gradient based methods [3–11]. More recently, extensions of these connections to nonsmooth settings using proximal methods have started to be considered [12–18]. Proximal algorithms play an important role in optimization since they can enjoy improved stability and be applied under weaker assumptions, and in many cases the associated proximal operators have simple closed form expressions. The majority of known proximal algorithms fall into the following types:

- forward-backward splitting [19–21];
- forward-backward-forward or Tseng splitting [22];
- Douglas-Rachford [19, 23];
- Davis-Yin three operator splitting [24]; and
- alternating direction method of multipliers (ADMM) [25, 26].

Many more sophisticated methods are extensions or variations of these themes. The first three were the only known methods for quite a long time. Only recently have Davis and Yin [24] solved the problem of obtaining a three operator splitting that cannot be reduced to any of the existing two operator splitting schemes. Such proximal methods are based on fixed-point iterations of nonexpansive monotone operators. A different technique based on projections onto separating sets has recently been proposed but will not be considered in this paper; see [27] and the references therein. ADMM dates back to the 70's and has gained popularity due to its effectiveness in solving large-scale problems with sparse and low rank regularizations [28].

We will focus on proximal algorithms that have previously been introduced from an *operator splitting* approach.[1] The literature on operator splitting is huge, so here we simply mention that these methods have origins in functional analysis and differential equations [29–31], and were later explored in convex analysis and optimization; see [32–37]. (See also [38, 39] for an introduction and historical account.)

1.1 Summary of main results

We consider algorithms for solving

$$\min_{x\in\mathbb{R}^n}\{F(x) \equiv f(x) + g(x) + w(x)\} \tag{1.1}$$

where f, g and w are functions from $\mathbb{R}^n$ into $\mathbb{R}$ obeying the following condition.

Assumption 1.1. *The functions f and g are proper, closed, and convex. The function w is differentiable.*

[1] One should not confuse operator splitting in convex analysis with splitting methods for ODEs.

This assumption will be used throughout the paper. The convexity requirements ensure that the proximal operators associated to f and g are well-defined, i.e., have a unique solution. For simplicity, in the discussion below we also assume that f and g are differentiable, however this condition can be relaxed.

Perhaps surprisingly, we show that *all* of the above mentioned (bulleted) proximal algorithms can be obtained as different discretizations (more precisely first-order integrators) of the simple *gradient flow* introduced by Cauchy [40] and given by

$$\dot{x} = -\nabla F(x), \tag{1.2}$$

where $x = x(t) \in \mathbb{R}^n$, t is the time variable, and $\dot{x} \equiv dx/dt$. Our approach makes connections between optimization algorithms with *ODE splitting methods* [41, 42], which is a powerful approach for designing algorithms. Interestingly, within this approach, ADMM emerges as a *rebalanced splitting* scheme, which is a recently introduced technique designed to preserve steady states of the underlying ODE [43]. We show that the dual variable associated with ADMM is precisely the *balance coefficient* used in this approach. We show that the other algorithms also preserve steady states of the gradient flow, but for different reasons, which in turn sheds light on the connections between ODE splitting ideas and operator splitting techniques from convex analysis.

The emphasis of this paper is on accelerated algorithms. We show that by employing similar discretization strategies to the *accelerated gradient flow* given by

$$\ddot{x} + \eta(t)\dot{x} = -\nabla F(x), \tag{1.3}$$

where $\ddot{x} \equiv d^2x/dt^2$ and $\eta(t)$ is a damping coefficient chosen to be either

$$\eta(t) = r/t \qquad \text{(decaying damping with } r \ge 3) \tag{1.4}$$

or

$$\eta(t) = r \qquad \text{(constant damping } r > 0), \tag{1.5}$$

one obtains *accelerated variants* of the respective algorithms. The vanishing damping (1.4) is related to the ODE from which Nesterov's accelerated gradient method may be obtained [3], while the constant damping (1.5) is related to the ODE associated with Polyak's heavy ball method [1]. We will refer to algorithms arising as discretizations of (1.3) derived with (1.4) as accelerated variants with "decaying damping," and those derived with (1.5) as accelerated variants with "constant damping." In our analysis we treat both types in a unified manner. We also note that choices other than (1.4) and (1.5) are possible and would be automatically covered by our framework. To the best of our knowledge, the accelerated frameworks that we introduce are new, although some recover known methods as special cases. We show that they are all *first-order integrators* that *preserve steady states* of the accelerated gradient flow (1.3).

We also show how this continuous-to-discrete analysis can be extended to monotone operators using two approaches: *differential inclusions* (nonsmooth dynamical systems) and the combination of ODEs and Yosida regularized operators.

	convex	strongly convex
gradient flow (1.2)	$\mathcal{O}\left(t^{-1}\right)$	$\mathcal{O}\left(e^{-mt}\right)$
accelerated gradient flow (1.4)	$\mathcal{O}\left(t^{-2}\right)$	$\mathcal{O}\left(t^{-2r/3}\right)$
accelerated gradient flow (1.5)	$\mathcal{O}\left(t^{-1}\right)$	$\mathcal{O}\big(e^{-\sqrt{m}t}\big)$

Table 1: Convergence rates of gradient flow (1.2) vs. accelerated gradient flow (1.3) with decaying damping (1.4) and constant damping (1.5). Let $F^\star \equiv \inf_{x\in\mathbb{R}^n} F(x)$ and $x^\star$ be the unique minimizer of an m-strongly convex function F. We show rates for $F(x(t)) - F^\star$ when F is convex, and for $\|x(t) - x^\star\|^2$ when F is strongly convex. Note the tradeoff between decaying and constant damping for convex vs. strongly convex functions. These rates follow as special cases of the analysis in [18].

Overall, this paper brings an alternative perspective on the design of proximal optimization algorithms by establishing tight relationships with continuous dynamical systems and numerical analyses of ODEs. Since the dynamical systems (1.2) and (1.3) play a central role in the paper, we summarize their convergence rates in Table 1. One expects that a suitable discretization would, at least up to a small error, preserve the same rates. However, a discrete analysis is still necessary to formalize such results.

1.2 Basic building blocks

Standard discretizations of the derivatives appearing in the first- and second-order systems (1.2) and (1.3), respectively, are

$$\dot{x}(t_k) = \pm(x_{k\pm1} - x_k)/h + \mathcal{O}(h), \tag{1.6}$$
$$\ddot{x}(t_k) = (x_{k+1} - 2x_k + x_{k-1})/h^2 + \mathcal{O}(h), \tag{1.7}$$

where $t_k = kh$ for $k \in \{0, 1, 2, \dots\}$, x_k is an approximation to the true trajectory $x(t_k)$, and $h > 0$ is the step size. To discretize (1.3) it will be convenient to define

$$\hat{x}_k \equiv x_k + \gamma_k(x_k - x_{k-1}) \quad \text{with} \quad \gamma_k = \begin{cases} \frac{k}{k+r} & \text{for decaying damping (1.4),} \\ 1 - rh & \text{for constant damping (1.5).} \end{cases} \tag{1.8}$$

Using this notation and the "minus" choice in (1.6), one can verify that

$$\ddot{x}(t_k) + \eta(t_k)\dot{x}(t_k) = (x_{k+1} - \hat{x}_k)/h^2 + \mathcal{O}(h). \tag{1.9}$$

As usual, we will often keep the leading order term and neglect $\mathcal{O}(h)$ terms. We note that although we consider (1.8), other choices of damping $\eta(t)$ in (1.3) are possible, which would lead to the same discretization (1.9) but with a different γ_k in (1.8).

Since this paper focuses on proximal methods, the *resolvent* operator plays a central role, and from a dynamical systems perspective so do *implicit* discretizations. For example, consider the implicit Euler discretization of (1.2) given by

$$(x_{k+1} - x_k)/h = -\nabla F(x_{k+1}). \tag{1.10}$$

This nonlinear equation in x_{k+1} can be solved using the *resolvent*,[2] which for an operator A and spectral parameter $\lambda \in \mathbb{C}$ is defined as

$$J_{\lambda A} \equiv (I + \lambda A)^{-1}. \tag{1.11}$$

We are interested in cases where A is a maximal monotone operator (see Section 4) and will always use $\lambda > 0$ as a real number related to the discretization step size. For instance, when $A = \nabla F$, $\lambda > 0$, and the *proximal operator* $\mathrm{prox}_{\lambda F}$ of F is well-defined, it follows that the resolvent is equal to[3]

$$J_{\lambda \nabla F}(v) \equiv \mathrm{prox}_{\lambda F}(v) \equiv \arg\min_x \left(F(x) + \tfrac{1}{2\lambda}\|x - v\|^2 \right). \tag{1.12}$$

It follows from (1.10), (1.11), and (1.12) that

$$x_{k+1} = J_{h\nabla F}(x_k) = \mathrm{prox}_{hF}(x_k), \tag{1.13}$$

which is the *proximal point algorithm* [44, 45] (see [36, 46, 47] for a thorough analysis and generalizations). When F is convex, the proximal point algorithm is known to converge when a minimizer exists. In practice, it is often more stable than gradient descent and allows for a more aggressive choice of step size, as is common for implicit discretizations. Although the proximal point algorithm has the cost of computing the operator in (1.12), it can be employed even when F is nondifferentiable.

The result below follows from (1.13) for the case of gradient flow, and from (1.8), (1.9), and (1.11) for the case of accelerated gradient flow. This lemma is a key building block for the design of the algorithmic frameworks in this paper.

Lemma 1.2. *An implicit Euler discretization of the gradient flow* (1.2) *and the accelerated gradient flow* (1.3) *with stepsize $h > 0$ yields, respectively, the updates*

$$x_{k+1} = \begin{cases} J_{h\nabla F}(x_k) & \textit{for the gradient flow (1.2)}, \\ J_{h^2\nabla F}(\hat{x}_k) & \textit{for the accelerated gradient flow (1.3)}, \end{cases} \tag{1.14}$$

where $\hat{x}_k$ is defined in (1.8) *based on which type of damping is used.*

[2] The resolvent was introduced by Fredholm in the late 19th century to study integral equations related to partial differential equations. This name was coined by Hilbert who used it extensively to develop the theory of linear operators. Usually, the resolvent is defined as $R(\lambda) \equiv (A - \lambda I)^{-1}$ when studying the spectral decomposition of A. However, in convex analysis the resolvent is defined as (1.11), but both are related via $J_{\lambda A} = \lambda^{-1} R(-\lambda^{-1})$.

[3] This holds for nondifferentiable functions as well where $A = \partial F$ is the subdifferential of F. In this case the differential equation (1.2) is replaced by a differential inclusion (see Section 4). Thus, although we often denote the proximal operator by $J_{\lambda \nabla F}$ due to the connection with ODEs, the reader should keep in mind that this applies to nonsmooth functions as well and the resulting algorithm does not require differentiability. Recall also that $\partial F(x) = \{\nabla F(x)\}$ when F is differentiable.

The update (1.14) is a proximal point computation associated with either x_k or $\hat{x}_k$, depending on whether acceleration is used. In the accelerated case, since these updates result from the discretization of (1.3), we obtain an intuitive interpretation: in the case of (1.8) the parameter r controls the amount of friction (dissipation) in the system, and for the first choice the friction is decaying over time, while for the second choice the friction is constant. Other choices are also possible.

1.3 Outline

In Section 2, we introduce the balanced and rebalanced splitting approaches recently proposed in [43]. However, we propose two modifications to this scheme that will enable us to make connections with existing optimization algorithms as well as propose new ones. In Section 3, we derive accelerated extensions of ADMM, accelerated extensions of the Davis-Yin method (forward-backward and Douglas-Rachford follow as special cases), and accelerated extensions of Tseng's method from an ODE splitting approach. We also show that all discretizations considered in this paper are proper first-order integrators that preserve steady states of the underlying ODE. In Section 4, we argue that our analysis extends to the nonsmooth setting, and to maximal monotone operators more generally. Finally, numerical results in Section 5 illustrate the speedup achieved by our accelerated variants.

2 Splitting Methods for ODEs

Assume that solving or simulating the ODE

$$\dot{x} = \varphi(x) \tag{2.1}$$

is an intractable problem, i.e., the structure of φ makes the problem not computationally amenable to an iterative numerical procedure. We denote the flow map of (2.1) by Φ_t. The idea is then to split the vector field $\varphi : \mathbb{R}^n \to \mathbb{R}^n$ into parts, each integrable or amenable to a feasible numerical approximation. For simplicity, consider

$$\varphi = \varphi^{(1)} + \varphi^{(2)} \tag{2.2}$$

and suppose that both

$$\dot{x} = \varphi^{(1)}(x), \qquad \dot{x} = \varphi^{(2)}(x), \tag{2.3}$$

are feasible, either analytically or numerically, with respective flow maps $\Phi_t^{(1)}$ and $\Phi_t^{(2)}$. For step size h, it can be shown that the simplest composition [48]

$$\hat{\Phi}_h = \Phi_h^{(2)} \circ \Phi_h^{(1)} \tag{2.4}$$

provides a first-order approximation in the following sense.

Definition 2.1. *Consider a map $\hat{\Phi}_h : \mathbb{R}^n \to \mathbb{R}^n$, with step size $h > 0$, which approximates the true flow Φ_t of the ODE* (2.1) *with vector field $\varphi : \mathbb{R}^n \to \mathbb{R}^n$. Then $\hat{\Phi}_h$ is said to be an integrator of order p if, for any $x \in \mathbb{R}^n$, it holds that*

$$\|\Phi_h(x) - \hat{\Phi}_h(x)\| = \mathcal{O}(h^{p+1}). \tag{2.5}$$

This implies a global error $\|\Phi_{t_k}(x) - (\hat{\Phi}_h)^k(x)\| = \mathcal{O}(h)$ for a finite interval $t_k = hk$.

There are different ways to compose individual flows, each resulting in a different method. For instance, one can use a preprocessor map $\chi_h : \mathbb{R}^n \to \mathbb{R}^n$ such that

$$\tilde{\Phi}_h = \chi_h^{-1} \circ \hat{\Phi}_h \circ \chi_h \tag{2.6}$$

is more accurate than $\hat{\Phi}_h$ with little extra cost. There are many interesting ideas in splitting methods for ODEs, some quite sophisticated. We mention these options to highlight that exploration beyond that considered in this paper is possible [41, 48, 49]. Naturally, more accurate methods are more expensive since they involve extra computation of individual flows. A good balance between accuracy and computational cost are methods of order $p = 2$. Here we will focus on the simple first-order scheme (2.4), which suffices to make connections with many optimization methods.

2.1 Balanced splitting

In general, splittings such as (2.2) do not preserve steady states of the system (2.1). Recently, an approach designed to preserve steady states was proposed under the name of *balanced splitting* [43]. The idea is to introduce a balance coefficient $c = c(t)$ into (2.1) by writing $\dot{x} = \varphi(x) + c - c$ and then perform a splitting as above, which results in the pair of ODEs

$$\dot{x} = \varphi^{(1)}(x) + c, \qquad \dot{x} = \varphi^{(2)}(x) - c. \tag{2.7}$$

We now show how this may be used to preserve steady states of the system (2.1).

First, assume x_∞ is a steady state of the system (2.1) so that $x_\infty = \lim_{t\to\infty} x(t)$ satisfies $\varphi^{(1)}(x_\infty) + \varphi^{(2)}(x_\infty) = 0$. If $c_\infty = \lim_{t\to\infty} c(t)$ is found to satisfy

$$c_\infty = \tfrac{1}{2}\big(\varphi^{(2)}(x_\infty) - \varphi^{(1)}(x_\infty)\big) \tag{2.8}$$

then x_∞ is also a stationary state for both ODEs in (2.7) since

$$\varphi^{(1)}(x_\infty) + c_\infty = \tfrac{1}{2}\big(\varphi^{(1)} + \varphi^{(2)}\big)(x_\infty) = 0, \tag{2.9a}$$

$$\varphi^{(2)}(x_\infty) - c_\infty = \tfrac{1}{2}\big(\varphi^{(2)} + \varphi^{(1)}\big)(x_\infty) = 0. \tag{2.9b}$$

To establish a result for the other direction, now assume that x_∞ is a steady state of both ODEs in (2.7). It follows that

$$c_\infty = \varphi^{(2)}(x_\infty) = -\varphi^{(1)}(x_\infty) = \tfrac{1}{2}\big(\varphi^{(2)}(x_\infty) - \varphi^{(1)}(x_\infty)\big). \tag{2.10}$$

From (2.10) and the fact that x_∞ is stationary for both systems in (2.7) gives

$$0 = \varphi^{(1)}(x_\infty) + c_\infty = \tfrac{1}{2}\big(\varphi^{(1)} + \varphi^{(2)}\big)(x_\infty), \tag{2.11a}$$
$$0 = \varphi^{(2)}(x_\infty) - c_\infty = \tfrac{1}{2}\big(\varphi^{(2)} + \varphi^{(1)}\big)(x_\infty), \tag{2.11b}$$

so that both equations in (2.11) imply that x_∞ is stationary for (2.1).

Motivated by (2.10), this can be implemented by computing the updates $c_k = \frac{1}{2}\big(\varphi^{(2)}(x_k) - \varphi^{(1)}(x_k)\big)$ during the numerical method, together with discretizations of the ODEs in (2.7). Note that this approach requires the explicit computation of $\varphi^{(i)}$. In optimization, one might have $\varphi^{(i)} = -\nabla f^{(i)}$, in which case it is not well-defined when $f^{(i)}$ is nonsmooth. We address this concern in the next section.

2.2 Rebalanced splitting

The *rebalanced splitting* approach was proposed by [43], and claimed to be more stable than the balanced splitting of Section 2.1. Importantly, for our purposes, it allows for the computation of a balance coefficient using only the previous iterates so that, in particular, no evaluation of $\varphi^{(i)}$ is needed.

Let $t_k = kh$ for $k \in \{0, 1, 2, \dots\}$ and step size $h > 0$. We then integrate $\dot{x} = \varphi^{(1)}(x) + c_k$ with initial condition $x(t_k) = x_k$ over the interval $[t_k, t_k + h]$ to obtain $x_{k+1/2}$, and then integrate $\dot{x} = \varphi^{(2)}(x) - c_k$ over the interval $[t_k, t_k + h]$ with initial condition $x(t_k) = x_{k+1/2}$ to obtain x_{k+1} (note that c_k is kept fixed during this procedure). The resulting integrals are given by

$$x_{k+1/2} = x_k + \int_{t_k}^{t_k+h} \big(\varphi^{(1)}(x(t)) + c_k\big) dt, \tag{2.12a}$$
$$x_{k+1} = x_{k+1/2} + \int_{t_k}^{t_k+h} \big(\varphi^{(2)}(x(t)) - c_k\big) dt. \tag{2.12b}$$

In light of (2.10), two reasonable ways of computing c_{k+1} are given by the *average* of either $\frac{1}{2}(\varphi^{(1)} - \varphi^{(2)})$ or $\varphi^{(2)}$ over the time step, which with (2.12) gives, respectively,

$$c_{k+1} = \frac{1}{h}\int_{t_k}^{t_k+h} \frac{\varphi^{(2)}(x(t)) - \varphi^{(1)}(x(t))}{2} dt = c_k + \frac{1}{h}\left(\frac{x_{k+1} + x_k}{2} - x_{k+1/2}\right), \tag{2.13a}$$
$$c_{k+1} = \frac{1}{h}\int_{t_k}^{t_k+h} \varphi^{(2)}(x(t)) dt = c_k + \frac{1}{h}\left(x_{k+1} - x_{k+1/2}\right). \tag{2.13b}$$

In contrast to the balanced case in Section 2.1, both of these options need not compute $\varphi^{(i)}$ to obtain c_{k+1} as shown in (2.13). Thus, the above approaches are better suited for nonsmooth optimization since they do not require explicit gradient computations.

Both options in (2.13) are slight variations of the approach proposed in [43]. To motivate the potential usefulness of (2.13b) compared to (2.13a), let us first remark that the updates to the balance coefficient play an important role in the stability of the numerical method [43]. For example, if $\varphi^{(2)}$ is much more "stiff" than $\varphi^{(1)}$, the method may be unstable for large step sizes. In an optimization context where $\varphi^{(2)}$ may be related to a regularizer (e.g., $\varphi^{(2)}(x) = \partial\|x\|_1$), it may be desirable to preserve steady states only through the first system in (2.7), which leads to the choice (2.13b). In Section 3.1, we show that this type of rebalanced splitting is related to the ADMM algorithm since the dual variable is precisely the balance coefficient in (2.13b).

3 Proximal Algorithms from ODE Splittings

We now use the previous ideas to construct implicit discretizations of both the gradient flow (1.2) and the accelerated gradient flow (1.3). However, our emphasis is on the latter since the analysis is more involved and can be easily adapted to the former. In addition to Assumption 1.1, throughout this section we assume the following conditions.

Assumption 3.1. *The functions f, g and w in the optimization problem* (1.1) *are continuous differentiable with Lipschitz continuous gradients.*

Lipschitz continuity of the gradients ensures uniqueness of solutions of the ODEs [50].

Finally, in continuous-time both the gradient flow (1.2) and the accelerated gradient flow (1.3), with damping as in (1.4) or (1.5), asymptotically solve the optimization problem (1.1) since these systems are stable and their trajectories tend to lower level sets of F [17,18]. In the following we construct suitable discretizations of these ODEs.

3.1 Accelerated extensions of ADMM

Let us introduce a balance coefficient $c = c(t)$ and write (1.3) as a first-order system:

$$\dot{x} = v, \qquad \dot{v} = \underbrace{-\eta(t)v - \nabla f(x) - \nabla w(x)}_{\varphi^{(1)}} \underbrace{-\nabla g(x)}_{\varphi^{(2)}} + c - c. \tag{3.1}$$

Splitting the second ODE above as indicated, we obtain the two independent systems

$$\begin{cases} \dot{x} &= v \\ \dot{v} &= -\eta(t)v - \nabla f(x) - \nabla w(x) + c, \end{cases} \qquad \begin{cases} \dot{x} &= v \\ \dot{v} &= -\nabla g(x) - c. \end{cases} \tag{3.2}$$

Note that we are only splitting the second equation in (3.1). It will be convenient to treat each of these respective systems in their equivalent second-order forms:

$$\ddot{x} + \eta(t)\dot{x} = -\nabla f(x) - \nabla w(x) + c, \qquad \ddot{x} = -\nabla g(x) - c. \tag{3.3}$$

We now discretize these systems using the results from Lemma 1.2, although we introduce some intermediary steps that will be justified by our analysis later. To this end, let us choose a step size parametrized as $h \equiv \sqrt{\lambda}$, and then use (1.9) (after dropping the $O(h)$ error term) and a semi-implicit discretization on the first equation of (3.3) to obtain the equation

$$x_{k+1/2} - \hat{x}_k = -\lambda \nabla f(x_{k+1/2}) - \lambda \nabla w(\hat{x}_k) + \lambda c_k. \tag{3.4}$$

This can now be solved with the resolvent (1.11) (i.e., in an analogous way as the second relation in (1.14)) to obtain the equation

$$x_{k+1/2} = J_{\lambda \nabla f}\big(\hat{x}_k - \lambda \nabla w(\hat{x}_k) + \lambda c_k\big). \tag{3.5}$$

For the second ODE in (3.3) we use (1.7) (after dropping the $O(h)$ term) and an implicit discretization to obtain the equation

$$\tilde{x}_{k+1} - 2x_{k+1/2} + \hat{x}_k = -\lambda \nabla g(x_{k+1}) - \lambda c_k \tag{3.6}$$

where

$$\tilde{x}_{k+1} = x_{k+1} + (x_{k+1/2} - \hat{x}_k). \tag{3.7}$$

Note that the endpoint $\tilde{x}_{k+1}$ is related to the other endpoint x_{k+1} via the momentum term $(x_{k+1/2} - \hat{x}_k)$ based on the first splitting, and together this results in[4]

$$x_{k+1} - x_{k+1/2} = -\lambda \nabla g(x_{k+1}) - \lambda c_k. \tag{3.8}$$

This implicit equation can again be solved with the resolvent (1.11) yielding

$$x_{k+1} = J_{\lambda \nabla g}\big(x_{k+1/2} - \lambda c_k\big). \tag{3.9}$$

For the balance coefficient c, we use the update (2.13b) based on $\varphi^{(2)} = -\nabla g$ (see (3.1)). An implicit discretization is equivalent to approximating the integral by its upper limit, which in this cases results in

$$c_{k+1} = \frac{1}{h}\int_{t_k}^{t_k+h} \varphi^{(2)}(x(t))dt = -\nabla g(x_{k+1}) + \mathcal{O}(h). \tag{3.10}$$

Using (3.8), and neglecting $\mathcal{O}(h)$ terms, we thus obtain

$$c_{k+1} = c_k + \lambda^{-1}\left(x_{k+1} - x_{k+1/2}\right). \tag{3.11}$$

Collecting the updates (3.5), (3.9), (3.11), and (1.8) we obtain Algorithm 1.

We would like to stress some important aspects of Algorithm 1.

[4] Note that $\tilde{x}_{k+1}$ in (3.7) is a little further away from x_{k+1}, which makes the algorithm "look ahead" and implicitly introduces dependency on the curvature of g in the resulting update.

Algorithm 1 Accelerated extension of ADMM for solving problem (1.1).

```
Choose $\lambda > 0$, and initialize $c_0$ and $\hat{x}_0$.
Choose $r \geq 3$ if decaying damping (1.4), or $r > 0$ if constant damping (1.5).
for $k = 0, 1, \ldots$ do
    $x_{k+1/2} \leftarrow J_{\lambda \nabla f}\left(\hat{x}_k - \lambda \nabla w(\hat{x}_k) + \lambda c_k\right)$
    $x_{k+1} \leftarrow J_{\lambda \nabla g}(x_{k+1/2} - \lambda c_k)$
    $c_{k+1} \leftarrow c_k + \lambda^{-1}\left(x_{k+1} - x_{k+1/2}\right)$
    Using $h = \sqrt{\lambda}$ and $r$, compute $\gamma_{k+1}$ and $\hat{x}_{k+1}$ from (1.8).
end for
```

- The standard ADMM [25, 26] is recovered from Algorithm 1 when $w = 0$ in problem (1.1) and no acceleration is used, i.e., when $\gamma_k = 0$ so that $\hat{x}_k = x_k$ for all k. Algorithm 1 extends ADMM to handle the case $w \neq 0$ in problem (1.1) by incorporating ∇w in the update to $x_{k+1/2}$ in (3.5).

- The dual vector update in ADMM is here represented by the update to the balance coefficient c_k, which as described earlier aims to preserve critical points of the underlying ODE. This brings new meaning to the dual vector.

- Acceleration through the update to $\hat{x}_k$ based on vanishing and constant damping in (1.8) have been considered. However, one is free to consider other damping functions $\eta(t)$ in the dynamical system (1.3) as well. By a suitable discretization, this would lead to a new update to γ_k in (1.8). For example, choosing $\eta(t) = r_1/t + r_2$ for constants $\{r_1, r_2\} \subset (0, \infty)$ yields
$$\gamma_k = k/(k + r_1) + r_2. \tag{3.12}$$
This observation is valid for every accelerated algorithm derived in this paper.

- When decaying damping is chosen in (1.8) and $w = 0$ in problem (1.1), Algorithm 1 is similar to the Fast ADMM proposed in [51]. They differ in that the latter also "accelerates" the dual variable c (i.e., the Lagrange multiplier update). Connections between Fast ADMM and continuous dynamical systems was recently considered in [17, 18] and corresponds to system (1.3) with $w = 0$. However, in this case the discretization is not a rebalanced splitting.

- The choice of discretization leading to (3.8), which involves relating $\tilde{x}_{k+1}$ to x_{k+1} (recall (3.7)), is motivated by obtaining updates similar to ADMM. This choice is formally justified by Theorem 3.3, which shows that the discretization has a local error of $\mathcal{O}(h^2)$ compared to the continuous trajectory.

Remark 3.2 (non-accelerated algorithms via gradient flow). Although we focus on accelerated algorithms, similar (and easier) analyses apply to the gradient flow (1.2) which lead to non-accelerated variants of the respective algorithm. For example, as in (3.1), one can introduce a balance coefficient c and split the system (1.2) into

$$\dot{x} = -\nabla f(x) - \nabla w(x) + c, \qquad \dot{x} = -\nabla g(x) - c. \tag{3.13}$$

Then, as for (3.4), a semi-implicit discretization of the first equation yields

$$x_{k+1/2} = J_{\lambda\nabla f}(x_k - \lambda\nabla w(x_k) + \lambda c_k), \tag{3.14}$$

where now $h \equiv \lambda$ in (1.6). An implicit discretization of the second equation yields $x_{k+1} - x_{k+1/2} = -\lambda\nabla g(x_{k+1}) - \lambda c_k$, so that x_{k+1} can be computed as

$$x_{k+1} = J_{\lambda\nabla g}(x_{k+1/2} - \lambda c_k). \tag{3.15}$$

The balance coefficient update (3.11) is obtained in an analogous manner as before. Note that updates (3.14) and (3.15), together with (3.11), are precisely the ADMM algorithm in the particular case $w = 0$.

The next result shows that the above discretization is justified since it yields a first-order integrator for the underlying ODE.

Theorem 3.3. *The following hold true:*

(i) Algo. 1 is a first-order integrator to the accelerated gradient flow (1.3)*;*

(ii) Algo. 1 with $\gamma_k = 0$ for all $k \geq 0$ is a first-order integrator to the gradient flow (1.2).

Proof. For f satisfying Assumption 1.1 and Assumption 3.1 it holds that $y = J_{\lambda\nabla f}(x)$ if and only if $y = x - \lambda\nabla f(y)$ (see (1.11)). Thus, Assumption 3.1 gives

$$y = J_{\lambda\nabla f}(x) = x - \lambda\nabla f(x - \lambda\nabla f(y)) = x - \lambda\nabla f(x) + \mathcal{O}(\lambda^2), \tag{3.16}$$

where the $\|\nabla f(y)\|$ that normally appears in the $\mathcal{O}$ term is suppressed because it is bounded independently of λ for all λ on a compact set as a consequence of $y = J_{\lambda\nabla f}(x)$ and Assumption 3.1. (A similar convention is used in (3.17) and (3.18) below.) Thus, this equality and a Taylor expansion on ∇f in the first update of Algo. 1 give

$$x_{k+1/2} = \hat{x}_k - \lambda\nabla w(\hat{x}_k) + \lambda c_k - \lambda\nabla f(\hat{x}_k) + \mathcal{O}(\lambda^2). \tag{3.17}$$

Similarly, the second update of Algo. 1 leads to

$$\begin{aligned} x_{k+1} &= x_{k+1/2} - \lambda c_k - \lambda\nabla g(x_{k+1/2}) + \mathcal{O}(\lambda^2) \\ &= \hat{x}_k - \lambda\nabla F(\hat{x}_k) + \mathcal{O}(\lambda^2) \end{aligned} \tag{3.18}$$

where to derive the second equality we used (3.17) and a Taylor expansion of ∇g. Recall (1.8) and note that $\gamma_k = 1 - \eta(t)h$ for constant damping ($\eta(t) = r$), while

$$\gamma_k = \frac{k}{k+r} = 1 - \frac{r}{k+r} = 1 - \frac{rh}{t_k}\left(1 + \frac{rh}{t_k}\right)^{-1} = 1 - \eta(t_k)h + \mathcal{O}(h^2) \tag{3.19}$$

for decaying damping ($\eta(t) = r/t$). Thus, in either case, we conclude that

$$\hat{x}_k = x_k + h\big(1 - \eta(t_k)h\big)v_k + \mathcal{O}(h^3) = x_k + \mathcal{O}(h) \tag{3.20}$$

where we have defined the velocity variable

$$v_k \equiv (x_k - x_{k-1})/h, \tag{3.21}$$

which is finite even in the limit $h \to 0$. Using (3.21), both equalities in (3.20), (3.18), and recalling that $\lambda \equiv h^2$, we conclude that

$$\begin{aligned} v_{k+1} &= v_k - h\eta(t_k)v_k - h\nabla F(x_k) + \mathcal{O}(h^2), \\ x_{k+1} &= x_k + hv_{k+1} = x_k + hv_k + \mathcal{O}(h^2). \end{aligned} \tag{3.22}$$

Now, consider the ODE (1.3), i.e., $\dot{x} = v$ and $\dot{v} = -\eta(t)v - \nabla F(x)$. Combining this with Taylor expansions we have

$$\begin{aligned} v(t+h) &= v(t) + h\dot{v}(t) + \mathcal{O}(h^2) = v(t) - h\eta(t)v(t) - h\nabla F(x(t)) + \mathcal{O}(h^2), \\ x(t+h) &= x(t) + h\dot{x}(t) + \mathcal{O}(h^2) = x(t) + hv(t) + \mathcal{O}(h^2). \end{aligned} \tag{3.23}$$

Therefore, by comparison with (3.22) we conclude that

$$v(t_{k+1}) = v_{k+1} + \mathcal{O}(h^2), \qquad x(t_{k+1}) = x_{k+1} + \mathcal{O}(h^2), \tag{3.24}$$

i.e., in one step of the algorithm the discrete trajectory agrees with the continuous trajectory up to $\mathcal{O}(h^2)$. This means that Definition 2.1 is satisfied with $p = 1$.

The above argument can be adapted to Algo. 1 with $\gamma_k = 0$ in relation to the gradient flow (1.2). The derivation is simpler and is thus omitted. □

3.2 Accelerated extensions of Davis-Yin

We now split the accelerated gradient flow (1.3) as in (3.1), but without introducing a balance coefficient, to obtain

$$\varphi^{(1)} = -\eta(t)v - \nabla f(x), \qquad \varphi^{(2)} = -\nabla g(x) - \nabla w(x). \tag{3.25}$$

Hence, instead of (3.3), we obtain the following two individual ODEs:

$$\ddot{x} + \eta(t)\dot{x} = -\nabla f(x), \qquad \ddot{x} = -\nabla g(x) - \nabla w(x). \tag{3.26}$$

An implicit discretization of the first system is

$$x_{k+1/4} - \hat{x}_k = -\lambda\nabla f(x_{k+1/4}) \tag{3.27}$$

Algorithm 2 Accelerated extension of Davis-Yin for solving problem (1.1).

```
Choose $\lambda > 0$, and initialize $\hat{x}_0$.
Choose $r \geq 3$ if decaying damping (1.4), or $r > 0$ if constant damping (1.5).
for $k = 0, 1, \ldots$ do
    $x_{k+1/4} \leftarrow J_{\lambda \nabla f}(\hat{x}_k)$
    $x_{k+1/2} \leftarrow 2x_{k+1/4} - \hat{x}_k$
    $x_{k+3/4} \leftarrow J_{\lambda \nabla g}\big(x_{k+1/2} - \lambda \nabla w(x_{k+1/4})\big)$
    $x_{k+1} \leftarrow \hat{x}_k + x_{k+3/4} - x_{k+1/4}$
    Using $h = \sqrt{\lambda}$ and $r$, compute $\gamma_{k+1}$ and $\hat{x}_{k+1}$ from (1.8).
end for
```

where $h \equiv \sqrt{\lambda}$, which as a result of Lemma 1.2 leads to

$$x_{k+1/4} \equiv \Phi_h^{(1)}(\hat{x}_k) = J_{\lambda \nabla f}(\hat{x}_k). \tag{3.28}$$

Next, to "inject momentum" in the direction of ∇f, we define the translation operator

$$\mathcal{T}_h(z) \equiv z - \lambda \nabla f(x_{k+1/4}) \tag{3.29}$$

for any vector z. Thus, the next point in the discretization is defined to be

$$x_{k+1/2} \equiv \mathcal{T}_h(x_{k+1/4}) = x_{k+1/4} - \lambda \nabla f(x_{k+1/4}) = 2x_{k+1/4} - \hat{x}_k \tag{3.30}$$

where we used (3.27) to obtain the last equality. Next, we can use (1.7) to obtain a semi-implicit discretization of the second system in (3.26) given by

$$x_{k+3/4} - 2x_{k+1/4} + \hat{x}_k = -\lambda \nabla g(x_{k+3/4}) - \lambda \nabla w(x_{k+1/4}). \tag{3.31}$$

This allows us to solve the implicit equation (3.31) in the form

$$x_{k+3/4} \equiv \Phi_h^{(2)}(\hat{x}_k) = J_{\lambda \nabla g}\left(x_{k+1/2} - \lambda \nabla w(x_{k+1/4})\right). \tag{3.32}$$

Finally, we apply the inverse $\mathcal{T}_h^{-1}(z) \equiv z + \lambda \nabla f(x_{k+1/4})$ and use (3.27) to obtain

$$x_{k+1} \equiv \mathcal{T}_h^{-1}(x_{k+3/4}) = x_{k+3/4} + \lambda \nabla f(x_{k+1/4}) = x_{k+3/4} - (x_{k+1/4} - \hat{x}_k). \tag{3.33}$$

The collection of (3.28), (3.30), (3.32), and (3.33) results in Algo. 2, and the entire discretization procedure is illustrated in Fig. 1.

The following comments concerning Algo. 2 are appropriate.

- Algo. 2 reduces to the Davis-Yin method [24] when $\gamma_k = 0$ in (1.8), so that $\hat{x}_k = x_k$ for all k, i.e., when no acceleration is used. In this case, it has been shown for convex functions that the method has a convergence result of $\mathcal{O}(1/k)$ in an average or ergodic sense, and when all functions are strongly convex and satisfy some regularity conditions that linear convergence holds [24]. In the non-accelerated case, the algorithm corresponds to the application of a similar discretization of the gradient flow (1.2) with splitting $\dot{x} = -\nabla f(x)$ and $\dot{x} = -\nabla g(x) - \nabla w(x)$ (also see Remark 3.2).

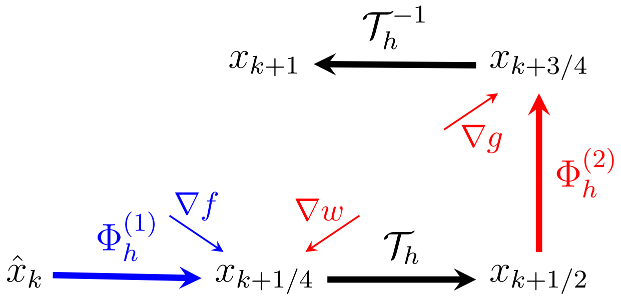

Figure 1: An illustration of the discretization underlying Algo. 2, consistent with the mapping (3.34), that gives the accelerated Davis-Yin method. We indicate the points at which the gradients act that define the implicit/explicit discretization.

- Algo. 2 is equivalent to the composition
$$\hat{\Phi}_h = \mathcal{T}_h^{-1} \circ \Phi_h^{(2)} \circ \mathcal{T}_h \circ \Phi_h^{(1)}. \tag{3.34}$$
Thus, comparing with (2.6) we see that $\mathcal{T}_h$ is actually a preprocessor map.
- In Theorem 3.4 we show that the above procedure yields a first-order integrator. Furthermore, in Theorem 3.6 we show that this discretization preserves critical points of the underlying ODE.

Theorem 3.4. *The following hold true:*

(i) Algo. 2 is a first-order integrator to the accelerated gradient flow (1.3)*;*

(ii) Algo. 2 with $\gamma_k = 0$ for all $k \geq 0$ is a first-order integrator to the gradient flow (1.2)*.*

Proof. The arguments are very similar to the proof of Theorem 3.3. From (3.16) and Taylor expansions, the first three updates of Algo. 2 yield

$$x_{k+1/4} = \hat{x}_k - \lambda \nabla f(\hat{x}_k) + \mathcal{O}(\lambda^2), \tag{3.35a}$$
$$x_{k+1/2} = \hat{x}_k - 2\lambda \nabla f(\hat{x}_k) + \mathcal{O}(\lambda^2), \tag{3.35b}$$
$$x_{k+3/4} = \hat{x}_k - 2\lambda \nabla f(\hat{x}_k) - \lambda \nabla g(\hat{x}_k) - \lambda \nabla w(\hat{x}_k) + \mathcal{O}(\lambda^2). \tag{3.35c}$$

Hence, the fourth update of Algo. 2 becomes

$$x_{k+1} = \hat{x}_k - \lambda \nabla F(\hat{x}_k) + \mathcal{O}(\lambda^2), \tag{3.36}$$

which is exactly the same as equation (3.18). Therefore, the remaining steps of the proof follow exactly as in the proof of Theorem 3.3, and establishes that Algo. 2 is an integrator with $p = 1$ according to Definition 2.1.

The proof related to the gradient flow (1.2) (i.e., with $\gamma_k = 0$ in Algo. 2) is similar, but easier, and therefore omitted. □

In order to show that Algo. 2 preserves critical points of the underlying ODE, we require the following technical result.

Lemma 3.5. *It holds that* $(\nabla f + \nabla g + \nabla w)(\bar{x}) = 0$ *if and only if* $\mathcal{P}(x) = x$ *with*

$$\mathcal{P} \equiv \tfrac{1}{2}I + \tfrac{1}{2}C_{\lambda\nabla g} \circ (C_{\lambda\nabla f} - \lambda\nabla w \circ J_{\lambda\nabla f}) - \tfrac{1}{2}\lambda\nabla w \circ J_{\lambda\nabla f}, \tag{3.37}$$

$\bar{x} = J_{\lambda\nabla f}(x)$*, and* $C_{\lambda\nabla f} \equiv 2J_{\lambda\nabla f} - I$ *is the Cayley operator.*

Proof. The first equation in the statement of the theorem is equivalent to $(I + \lambda\nabla g)(\bar{x}) = (I - \lambda\nabla f - \lambda\nabla w)(\bar{x})$ since $\lambda > 0$. Using the resolvent (1.11) this yields

$$\bar{x} = J_{\lambda\nabla g} \circ (I - \lambda\nabla f - \lambda\nabla w)(\bar{x}). \tag{3.38}$$

Now, making use of the identity

$$C_{\lambda\nabla f} \circ (I + \lambda\nabla f) = \big(2(I + \lambda\nabla f)^{-1} - I\big) \circ (I + \lambda\nabla f) = I - \lambda\nabla f \tag{3.39}$$

and substituting $J_{\lambda\nabla g} = \frac{1}{2}(C_{\lambda\nabla g} + I)$ into (3.38) we obtain

$$\bar{x} = \tfrac{1}{2}(C_{\lambda\nabla g} + I) \circ \{C_{\lambda\nabla f} \circ (I + \lambda\nabla f) - \lambda\nabla w\}(\bar{x}). \tag{3.40}$$

Since $x \equiv (I + \lambda\nabla f)(\bar{x})$, or equivalently $\bar{x} = J_{\lambda\nabla f}(x)$, it follows from (3.40) that

$$J_{\lambda\nabla f}(x) = \tfrac{1}{2}C_{\lambda\nabla g} \circ (C_{\lambda\nabla f} - \lambda\nabla w \circ J_{\lambda\nabla f})(x) + \tfrac{1}{2}C_{\lambda\nabla f}(x) - \tfrac{1}{2}\lambda\nabla w \circ J_{\lambda\nabla f}(x), \tag{3.41}$$

which by the definition of the Cayley operator is equivalent to

$$\tfrac{1}{2}x = \tfrac{1}{2}C_{\lambda\nabla g} \circ (C_{\lambda\nabla f} - \lambda\nabla w \circ J_{\lambda\nabla f})(x) - \tfrac{1}{2}\lambda\nabla w \circ J_{\lambda\nabla f}(x). \tag{3.42}$$

Adding $x/2$ to each side of the previous equality yields $x = \mathcal{P}(x)$, as claimed. □

Theorem 3.6. *If the iterate sequence* $\{x_k\}$ *generated by Algo. 2 satisfies* $\{x_k\} \to x_\infty$ *for some* x_∞*, then the vector* $\bar{x}_\infty \equiv J_{\lambda\nabla f}(x_\infty)$ *satisfies*

$$\{x_{k+1/4}\} \to \bar{x}_\infty, \qquad (\nabla f + \nabla g + \nabla w)(\bar{x}_\infty) = 0, \tag{3.43}$$

i.e., $\bar{x}_\infty$ *is a solution of* (1.1) *and a steady state of the accelerated gradient flow* (1.3)*. When* $\gamma_k = 0$ *for all* $k \geq 0$*,* $\bar{x}_\infty$ *is a steady state of the gradient flow* (1.2)*.*

Proof. It follows from the updates in Algo. 2, the definition of $C_{\lambda\nabla f}$ in the statement of Lemma 3.5, and the definition of $\mathcal{P}$ in (3.37) that $x_{k+1} = \hat{\Phi}_h(\hat{x}_k)$ where

$$\begin{aligned}
\hat{\Phi}_h &= I + J_{\lambda\nabla g} \circ (2J_{\lambda\nabla f} - I - \lambda\nabla w \circ J_{\lambda\nabla f}) - J_{\lambda\nabla f} \\
&= I + J_{\lambda\nabla g} \circ (C_{\lambda\nabla f} - \lambda\nabla w \circ J_{\lambda\nabla f}) - \tfrac{1}{2}(C_{\lambda\nabla f} + I) \\
&= \tfrac{1}{2}I - \tfrac{1}{2}C_{\lambda\nabla f} + \tfrac{1}{2}(C_{\lambda\nabla g} + I) \circ (C_{\lambda\nabla f} - \lambda\nabla w \circ J_{\lambda\nabla f}) \\
&= \tfrac{1}{2}I + \tfrac{1}{2}C_{\lambda\nabla g} \circ (C_{\lambda\nabla f} - \lambda\nabla w \circ J_{\lambda\nabla f}) - \tfrac{1}{2}\lambda\nabla w \circ J_{\lambda\nabla f} \\
&= \mathcal{P},
\end{aligned} \tag{3.44}$$

i.e., that $x_{k+1} = \mathcal{P}(\hat{x}_k)$. (Note that $\mathcal{P}$ is the operator $\mathcal{T}$ studied in [24] and proved to be an "averaged operator", which means that $\mathcal{P}$ is a nonexpansive operator and thus continuous [38, Remark 4.34].) Combining this with (1.8) shows that

$$\hat{x}_{k+1} = x_{k+1} + \gamma_{k+1}(x_{k+1} - x_k) = \mathcal{P}(\hat{x}_k) + \gamma_{k+1}(x_{k+1} - x_k). \tag{3.45}$$

By assumption there exists some x_∞ such that $\{x_k\} \to x_\infty$, which combined with (1.8) shows that $\{\hat{x}_k\} \to x_\infty$. Combining these facts with (3.45) and continuity of $\mathcal{P}$ establishes that $\mathcal{P}(x_\infty) = x_\infty$. It now follows from Lemma 3.5 that $\bar{x}_\infty = J_{\lambda\nabla f}(x_\infty)$ satisfies $(\nabla f + \nabla g + \nabla w)(\bar{x}_\infty) = 0$. Moreover, from the update for $x_{k+1/4}$ in Algo. 2 and continuity of $J_{\lambda\nabla f}$ we can conclude that

$$\lim_{k\to\infty} x_{k+1/4} = \lim_{k\to\infty} J_{\lambda\nabla f}(\hat{x}_k) = J_{\lambda\nabla f}(x_\infty) = \bar{x}_\infty. \tag{3.46}$$

This completes the proof for this case once we recall that $(\nabla f + \nabla g + \nabla w)(\bar{x}_\infty) = 0$.

The same argument applies when $\gamma_k = 0$ (i.e., when $\hat{x}_k = x_k$ for all $k \geq 0$), in which case Algo. 2 is a discretization of (1.2). □

3.2.1 Accelerated extensions of Douglas-Rachford

When Algo. 2 with $\gamma_k = 0$ for all k is applied to problem (1.1) with $w = 0$, one obtains the well-known Douglas-Rachford algorithm [19, 23], which has been extensively studied in the literature (for recent results see [52] and the references therein). Therefore, Douglas-Rachford is a discretization of the gradient flow (1.2) with $w = 0$. Also, from Algo. 2 one obtains accelerated variants of Douglas-Rachford that are discretizations of the accelerated gradient flow (1.3). For instance, when decaying damping in (1.8) is used, the resulting method was studied in [53]. We are not aware of previous work on accelerated variants with constant damping, or other choices such as (3.12).

3.2.2 Accelerated extensions of forward-backward

When Algo. 2 with $\gamma_k = 0$ for all k is applied to problem (1.1) with $f = 0$, one obtains the forward-backward splitting method [19–21], i.e., $x_{k+1} = J_{\lambda\nabla g}(x_k - \lambda\nabla w(x_k))$. This corresponds to a first-order integrator to the gradient flow (1.2). Similarly, for nonzero γ_k, Algo. 2 gives accelerated variants of forward-backward, i.e., $x_{k+1} = J_{\lambda\nabla g}\big(\hat{x}_k - \lambda\nabla w(\hat{x}_k)\big)$. From an ODE perspective, this is not a splitting method but rather a semi-implicit discretization; the first equation in (3.26) is absent (also see Fig. 2 (left) for an illustration). In any case, Theorem 3.4 ensures that the iterates correspond to first-order integrators, and Theorem 3.6 shows that critical points are preserved, where the operator in (3.44) reduces to $J_{\lambda\nabla g} \circ (I - \lambda\nabla w)$.

Algorithm 3 Accelerated Tseng's method for solving problem (1.1) when $f = 0$.

```
Choose $\lambda > 0$, and initialize $\hat{x}_0$.
Choose $r \geq 3$ if decaying damping (1.4), or $r > 0$ if constant damping (1.5).
for $k = 0, 1, \ldots$ do
    $x_{k+1/2} \leftarrow J_{\lambda \nabla g}\left(\hat{x}_k - \lambda \nabla w(\hat{x}_k)\right)$
    $x_{k+1} \leftarrow x_{k+1/2} - \lambda\left(\nabla w(x_{k+1/2}) - \nabla w(\hat{x}_k)\right)$
    Using $h = \sqrt{\lambda}$ and $r$, compute $\gamma_{k+1}$ and $\hat{x}_{k+1}$ from (1.8).
end for
```

3.3 Accelerated extensions of Tseng's splitting

The final proximal-based method to be considered is the forward-backward-forward splitting proposed by Tseng [22], which consists of a modification (a perturbation) of the forward-backward splitting discussed above. In order to propose accelerated extensions of Tseng's scheme, we consider the accelerated gradient flow (1.3) with $f = 0$ written as

$$\dot{x} = v, \qquad \dot{v} = \underbrace{-\eta(t)v - \nabla g(x) - \nabla w(x)}_{\varphi^{(1)}} + \underbrace{\nabla w(x) - \nabla w(x)}_{\varphi^{(2)}}. \tag{3.47}$$

Splitting this system leads to the two independent ODEs

$$\ddot{x} + \eta(t)\dot{x} = -\nabla g(x) - \nabla w(x), \qquad \ddot{x} = \nabla w(x) - \nabla w(x). \tag{3.48}$$

Using $h \equiv \sqrt{\lambda}$, (1.9), and a forward-backward discretization of the first equation gives

$$x_{k+1/2} = J_{\lambda \nabla g}\left(\hat{x}_k - \lambda \nabla w(\hat{x}_k)\right). \tag{3.49}$$

This is the same step as in the forward-backward method. For the second equation in (3.48) we use (1.7) in the form $\tilde{x}_{k+1} - 2x_{k+1/2} + \hat{x}_k = \lambda \nabla w(\hat{x}_k) - \lambda \nabla w(x_{k+1/2})$, where $\tilde{x}_{k+1}$ is given by (3.7). This gives

$$x_{k+1} = x_{k+1/2} - \lambda\big(\nabla w(x_{k+1/2}) - \nabla w(\hat{x}_k)\big). \tag{3.50}$$

By combining (3.49) and (3.50) we arrive at Algo. 3 (also see Fig. 2).

The original method proposed in [22] is recovered from Algo. 3 by setting $\gamma_k = 0$ for all k (i.e., without acceleration), in which case the algorithm is a discretization of the gradient flow (1.2). We believe that the accelerated variants in Algo. 3 have not previously been considered in the literature.

The next result shows that Algo. 3 is a first-order integrator.

Theorem 3.7. *The following hold true:*

(i) Algo. 3 is a first-order integrator to the accelerated gradient flow (1.3)*;*

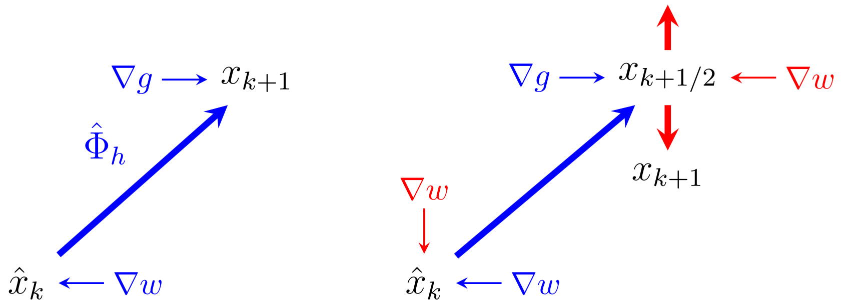
$\nabla g \longrightarrow x_{k+1}$
$\hat{\Phi}_h$
$\hat{x}_k \longleftarrow \nabla w$
$\nabla g \longrightarrow x_{k+1/2} \longleftarrow \nabla w$
x_{k+1}
∇w
$\hat{x}_k \longleftarrow \nabla w$

Figure 2: *Left:* Illustration of the accelerated forward-backward method, which is a semi-implicit Euler discretization. *Right:* Illustration of accelerated Tseng splitting, which adds a perturbation to the forward-backward method (see (3.50)).

(ii) Algo. 3 with $\gamma_k = 0$ for all $k \geq 0$ is a first-order integrator to the gradient flow (1.2)*.*

Proof. The results may be proved using the similar arguments as those used to establish Theorem 3.3 and Theorem 3.4. □

Theorem 3.8. *Let $f = 0$. If λ is sufficiently small and the iterate sequence $\{x_k\}$ generated by Algo. 3 satisfies $\{x_k\} \to x_\infty$ for some x_∞, then*

$$(\nabla g + \nabla w)(x_\infty) = 0, \tag{3.51}$$

i.e., x_∞ is a solution of problem (1.1) *and a steady state of the accelerated gradient flow* (1.3)*. When $\gamma_k = 0$ for all $k \geq 0$, x_∞ is a steady state of the gradient flow* (1.2)*.*

Proof. From the updates in Algo. 3, it follows that $x_{k+1} = \hat{\Phi}_h(\hat{x}_k)$ where

$$\hat{\Phi}_h = (I - \lambda \nabla w) \circ J_{\lambda \nabla g} \circ (I - \lambda \nabla w) + \lambda \nabla w. \tag{3.52}$$

Combining this with (1.8) shows that

$$\hat{x}_{k+1} = x_{k+1} + \gamma_{k+1}(x_{k+1} - x_k) = \hat{\Phi}_h(\hat{x}_k) + \gamma_{k+1}(x_{k+1} - x_k). \tag{3.53}$$

Combining $\{x_k\} \to x_\infty$ with the first equality in (3.53) and (1.8) yields $\{\hat{x}_k\} \to x_\infty$. Combining these facts with (3.53) and continuity of $\hat{\Phi}_h$ shows that

$$x_\infty = \lim_{k \to \infty} \hat{\Phi}_h(\hat{x}_k) = \hat{\Phi}_h(x_\infty). \tag{3.54}$$

Subtracting $\lambda \nabla w(x_\infty)$ from both sides of the previous inequality gives

$$(I - \lambda \nabla w)(x_\infty) = (I - \lambda \nabla w) \circ J_{\lambda \nabla g} \circ (I - \lambda \nabla w)(x_\infty). \tag{3.55}$$

Next, applying the operator $(I - \lambda \nabla w)^{-1}$, which exists for λ sufficiently small, yields

$$x_\infty = J_{\lambda \nabla g} \circ (I - \lambda \nabla w)(x_\infty), \tag{3.56}$$

which is itself equivalent to

$$(I + \lambda \nabla g)(x_\infty) = (I - \lambda \nabla w)(x_\infty). \tag{3.57}$$

Since $\lambda > 0$, the previous inequality shows that $(\nabla g + \nabla w)(x_\infty) = 0$, as claimed.

The same argument applies when $\gamma_k = 0$ (i.e., when $\hat{x}_k = x_k$ for all $k \geq 0$), in which case Algo. 3 is a discretization of (1.2). □

4 Monotone Operators

As indicated by Assumption 3.1, all previously considered operators associated with the resolvent (1.11) were single-valued. Since proximal algorithms can be generalized to the more abstract level of monotone operators, in this section we discuss how our previous analysis applies in this context.

Let us recall some concepts about monotone operators (see [38] for more details). Let H be a Hilbert space with inner product $\langle \cdot, \cdot \rangle : H \times H \to \mathbb{C}$. A multi-valued map $A : H \rightrightarrows H$ with $\operatorname{dom} A \equiv \{x \in H \,|\, Ax \neq \emptyset\}$ is *monotone* if and only if

$$\langle Ay - Ax, y - x \rangle \geq 0 \qquad \text{for all } x, y \in \operatorname{dom} A. \tag{4.1}$$

A monotone operator is said to be *maximal* if no enlargement of its graph is possible. Every monotone operator admits a maximal extension. Hence, from now on, every operator A is assumed to be maximal monotone. The resolvent of A with parameter λ is defined by (1.11) and can be shown to be a single-valued map, i.e., $J_{\lambda A} : H \to H$. Moreover, $x^\star \in \operatorname{zer}(A) \equiv \{x \in H \,|\, 0 \in Ax\}$ if and only if $J_{\lambda A}(x^\star) = x^\star$.

An important concept is the *Yosida regularization* of A with parameter $\mu > 0$:

$$A_\mu \equiv \mu^{-1}(I - J_{\mu A}). \tag{4.2}$$

This operator is *single-valued*, since $J_{\mu A}$ is single-valued, and Lipschitz continuous. Moreover, $0 \in Ax^\star$ if and only if $0 = A_\mu x^\star$, thus A and A_μ have the same zeros. It can be shown that in the limit $\mu \downarrow 0$ one has $A_\mu x \to A_0 x$, where $A_0 x \in Ax$ is the element of minimal norm. Often, one is faced with the resolvent of the Yosida regularization $J_{\lambda A_\mu} = (I + \lambda A_\mu)^{-1}$, which can be expressed in terms of the operator A by using

$$J_{\lambda A_\mu} = (\mu + \lambda)^{-1} \left(\mu I + \lambda J_{(\mu+\lambda)A} \right). \tag{4.3}$$

Importantly, in the limit $\mu \downarrow 0$ we see that (4.3) recovers the resolvent $J_{\lambda A}$.

4.1 Differential inclusions

Consider the following two differential inclusions (we refer to [54] for background on nonsmooth dynamical systems):

$$\dot{x} \in -Ax - Bx - Cx, \tag{4.4}$$

$$\ddot{x} + \eta(t)\dot{x} \in -Ax - Bx - Cx, \tag{4.5}$$

with $\eta(t)$ given by (1.4) or (1.5), and under the following assumption.

Assumption 4.1. *The operators $A, B : H \rightrightarrows H$ are maximal monotone. The operator $C : H \to H$ is maximal monotone and single-valued.*

Under Assumption 4.1, the differential inclusions (4.4) and (4.5) have a unique solution [54]. The previous discretizations of the gradient flow (1.2) and the accelerated gradient flow (1.3) extend naturally to (4.4) and (4.5), respectively. This is a consequence of the resolvent (1.11) being a single-valued map, which we illustrate through an example. Consider the procedure of Section 3.2 that led to Algo. 2. Using a similar procedure, we use a splitting of (4.5) to obtain the differential inclusions

$$\ddot{x} + \eta(t)\dot{x} \in -Ax, \qquad \ddot{x} + Cx \in -Bx. \tag{4.6}$$

An implicit discretization of the first inclusion yields $(I + \lambda A)(x_{k+1/4}) \ni \hat{x}_k$, while a semi-implicit discretization of the second inclusion together with the definition (3.30) yield $(I + \lambda B)(x_{k+3/4}) \ni x_{k+1/2} - \lambda C x_{k+1/4}$. Since under Assumption 4.1 the resolvents of A and B are single-valued, we can invert these relations to obtain

$$x_{k+1/4} = J_{\lambda A}(\hat{x}_k), \tag{4.7a}$$

$$x_{k+1/2} = 2x_{k+1/4} - \hat{x}_k, \tag{4.7b}$$

$$x_{k+3/4} = J_{\lambda B}(x_{k+1/2} - \lambda C x_{k+1/4}), \tag{4.7c}$$

$$x_{k+1} = x_{k+3/4} - (x_{k+1/4} - \hat{x}_k), \tag{4.7d}$$

$$\hat{x}_{k+1} = x_{k+1} + \gamma_k(x_{k+1} - x_k), \tag{4.7e}$$

where the second to last update follows from (3.33), and the last update from (1.8). The algorithm given by updates (4.7) is the "operator analog" of Algo. 2 that aims to find a vector x^* satisfying $0 \in (A + B + C)(x^\star)$.

Setting $C = 0$ into (4.7) one obtains the operator analog of the accelerated Douglas-Rachford (see Section 3.2.1), while setting $A = 0$ one obtains the operator analog of the accelerated forward-backward method (see Section 3.2.2). Operator analogs of Algo. 1 and Algo. 3 follow in a similar manner. One can also remove acceleration by setting $\gamma_k = 0$ for all $k \geq 0$, in which case these algorithms are discretizations of the first-order differential inclusion (4.4) (also see Remark 3.2).

Remark 4.2. We mention a subtlety regarding the order of accuracy of a discretization such as (4.7) to a differential inclusion such as (4.4) or (4.5). In the smooth case we concluded that such a discretization is a first-order integrator; see Definition 2.1 and Theorems 3.3, 3.4, and 3.7. An important ingredient was the Taylor approximation of the resolvent (3.16). However, for a maximal monotone operator A only the following weaker approximation is available [38, Remark 23.47]:

$$J_{\lambda A} = I - \lambda A_0 + o(\lambda) \tag{4.8}$$

where the action of $A_0 = \lim_{\mu \downarrow 0} A_\mu$ on a vector $x \in \operatorname{dom}(A)$ gives the minimal norm element of the set Ax. Thus, by the same argument used in the proofs of Theorems 3.3 and 3.4, but now using (4.8) and assuming that one can expand $A_0(x + \mathcal{O}(\lambda)) = A_0(x) + \mathcal{O}(\lambda)$ and similarly for B_0 and C_0, one concludes that the discrete and continuous trajectories agree up to $o(h)$. This is in contrast with the $\mathcal{O}(h^2)$ approximation that we proved for the smooth setting considered in Section 3.

4.2 Regularized ODEs

There is an alternative to considering the differential inclusions (4.4) and (4.5), which is to consider the respective ODEs

$$\dot{x} = -A_\mu x - B_\mu x - Cx, \tag{4.9}$$

$$\ddot{x} + \eta(t)\dot{x} = -A_\mu x - B_\mu x - Cx, \tag{4.10}$$

with the multivalued operators replaced by their respective Yosida regularization (4.2), which are single-valued and Lipschitz continuous. Thus, both ODEs admit a unique global solution. Note, however, that

$$\operatorname{zer}(A + B + C) \neq \operatorname{zer}(A_\mu + B_\mu + C) \tag{4.11}$$

so that the ODEs (4.9) and (4.10) do not have steady states that are compatible with zeros of the operator sum $A+B+C$. Nevertheless, after discretizing these regularized ODEs, one can take the limit $\mu \downarrow 0$ to recover iterates aimed at finding zeros of $A+B+C$. We stress that this procedure will give exactly the same updates as if one discretizes the original differential inclusions because of the identity (4.3); let us illustrate with an example. Consider the discretization procedure of Section 3.2 but applied to the ODE (4.10). Similarly to (3.28)–(3.33), together with the accelerated variable $\hat{x}_k$ in (1.8), we immediately obtain with the help of (4.3) the updates

$$x_{k+1/4} = (\mu + \lambda)^{-1}(\mu I + \lambda J_{(\mu+\lambda)A})(\hat{x}_k), \tag{4.12a}$$

$$x_{k+1/2} = 2x_{k+1/4} - \hat{x}_k, \tag{4.12b}$$

$$x_{k+3/4} = (\mu + \lambda)^{-1}(\mu I + \lambda J_{(\mu+\lambda)B})(x_{k+1/2} - \lambda C x_{k+1/4}), \tag{4.12c}$$

$$x_{k+1} = x_{k+3/4} - (x_{k+1/4} - \hat{x}_k), \tag{4.12d}$$

$$\hat{x}_{k+1} = x_k + \gamma_k(x_{k+1} - x_k). \tag{4.12e}$$

Taking the limit $\mu \downarrow 0$ above results in the updates (4.7), which we can recall as a discretization of the differential inclusion (4.5).

Remark 4.3. It is possible to generalize Lemma 3.5 to general maximal monotone operators, i.e., $0 \in (A + B + C)(\bar{x})$ if and only if $\mathcal{P}(x) = x$ where

$$\mathcal{P} \equiv \tfrac{1}{2}I + \tfrac{1}{2}C_{\lambda B} \circ (C_{\lambda A} - \lambda C \circ J_{\lambda A}) - \tfrac{1}{2}\lambda C \circ J_{\lambda A} \tag{4.13}$$

and $\bar{x} = J_{\lambda A}x$. The proof is similar to the one of Lemma 3.5, although one needs to be careful in replacing equalities by appropriate inclusions and using the fact that the resolvent of a maximal monotone operator is single-valued. Therefore, by the same arguments as in Theorem 3.6, the updates (4.7) preserve zeros of $A + B + C$.

Remark 4.4. To place the above concepts within an optimization context, consider the case where $A = \partial f$ is the subdifferential of a nonsmooth convex function f. Then, the Yosida regularization (4.3) becomes

$$\nabla f_\mu(x) = \mu^{-1}(x - \operatorname{prox}_{\mu f}(x)), \tag{4.14}$$

which is the gradient of the Moreau envelope $f_\mu(x) \equiv \min_y \big(f(y) + \frac{1}{2\mu}\|y - x\|^2\big)$. The Moreau envelope is always differentiable and has the same minimizers as f.

5 Numerical Experiments

Based on the continuous rates of Table 1, we expect that the accelerated algorithms that we have introduced will converge faster than their non-accelerated counterparts. In this section we investigate this numerically.

5.1 LASSO

Consider the well-known LASSO regression problem

$$\min_{x \in \mathbb{R}^n} \big\{ F(x) = \tfrac{1}{2}\|Ax - b\|^2 + \alpha\|x\|_1 \big\} \tag{5.1}$$

where $A \in \mathbb{R}^{m \times n}$ is a given matrix, $b \in \mathbb{R}^m$ is a given signal, and $\alpha > 0$ is a weighting parameter. We generate data by sampling $A \sim \mathcal{N}(0, 1)$, where $\mathcal{N}(\mu, \sigma)$ denotes a normal distribution with mean μ and standard deviation σ, and then normalizing its columns to have unit two norm. We sample $x_\bullet \in \mathbb{R}^n \sim \mathcal{N}(0, 1)$ with sparsity level 95% (i.e., only 5% of its entries are nonzero), and then add noise to obtain the observed signal $b = Ax_\bullet + e$, where the entries of e are chosen i.i.d. from a normal distribution with mean zero and standard deviation 10^{-3}. We choose $m = 500$ and $n = 2500$. The resulting signal-to-noise ratio is on the order of 250, and $x_\bullet$ has 125 nonzero entries. The parameter α is set as $\alpha = 0.1\alpha_{\max}$ where

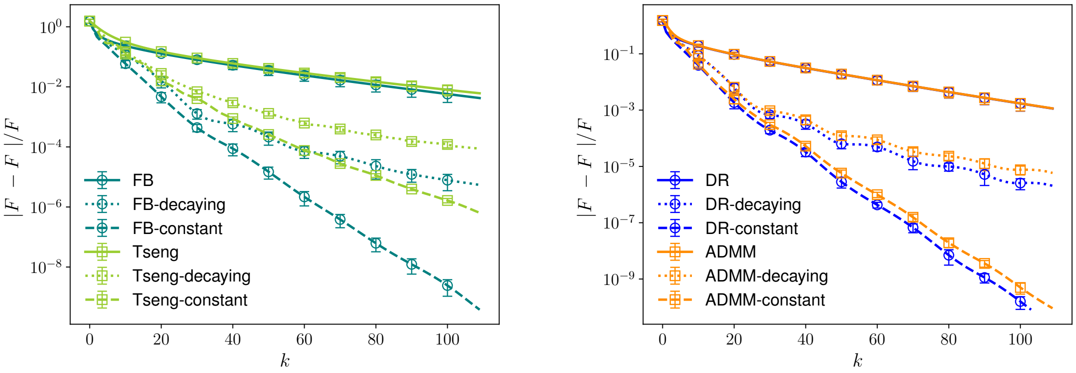

Figure 3: Performance of our twelve tested algorithm variants on problem (5.1). We perform 10 Monte-Carlo runs and show the mean and standard deviation of the relative error between $F_k = F(x_k)$ and F^*, where $F^\star$ is the solution obtained by CVXPY.

$\alpha_{\max} = \|A^T b\|_\infty$ is the maximum value for α such that (5.1) admits a nontrivial solution. We evaluate methods by computing the relative error $|F_k - F^\star|/F^\star$ where $F_k \equiv F(x_k)$ and $F^\star$ is the value of the objective obtained with the default implementation of CVXPY.

We compare four frameworks: ADMM and Douglas-Rachford (DR) (these correspond to Algo. 1 and Algo. 2, respectively, with $f(x) = \frac{1}{2}\|Ax - b\|_2^2$, $g(x) = \alpha\|x\|_1$, and $w(x) = 0$), and forward-backward (FB) splitting and Tseng splitting (these correspond to Algo. 2 and Algo. 3, respectively, with $f(x) = 0$, $g(x) = \alpha\|x\|_1$, $w(x) = \frac{1}{2}\|Ax - b\|_2^2$). For each of these four frameworks, we consider three variants: no acceleration (i.e., setting $\gamma_k = 0$ for all k), acceleration based on decaying damping (i.e., setting γ_k by (1.8) with damping coefficient defined in (1.4)), and acceleration based on constant damping (i.e., setting γ_k by (1.8) with damping coefficient defined in (1.5)). This results in a total of twelve algorithms that we denote by ADMM, ADMM-decaying, ADMM-constant, DR, DR-decaying, DR-constant, FB, FB-decaying, FB-constant, Tseng, Tseng-decaying, Tseng-constant. In all cases, we choose a step size of $\lambda = 0.1$ for the proximal operators. For decaying damping we choose $r = 3$, and for constant damping $r = 0.5$.

In Fig. 3 we report the mean and standard deviation (errorbars) across 10 randomly generated instances of problem (5.1) for various methods. The figure shows that the accelerated variants of each method improve over the non-accelerated variant. In particular, the constant damping accelerated variant is the fastest in this example.

5.2 Nonnegative matrix completion

We now consider a matrix completion problem where the entries of the matrix are constrained to lie in a specified range. Suppose that for a low-rank matrix $M \in \mathbb{R}^{n\times m}$, we only have access to certain entries whose ordered pairs are collected in a set Ω; let the operator $\mathcal{P}_\Omega : \mathbb{R}^{n\times m} \to \mathbb{R}^{n\times m}$ denote the projection onto these observable entries. The observable data matrix is given by $M_{\text{obs}} = \mathcal{P}_\Omega(M)$ where $[\mathcal{P}_\Omega(M)]_{ij} = M_{ij}$ if $(i,j) \in \Omega$ and $[\mathcal{P}_\Omega(M)]_{ij} = 0$ otherwise. The goal is then to estimate the missing entries of M. One popular approach is to solve the convex optimization problem $\min\{\|X\|_* \,|\, \mathcal{P}_\Omega(X) = \mathcal{P}_\Omega(M)\}$, where $\|X\|_*$ is the nuclear norm of X [55]. We consider a modification of this approach by imposing constraints of the form $a \le X_{ij} \le b$ for given constants a and b. Specifically, we solve

$$\min_{X \in \mathbb{R}^{n\times m}} \left\{ F(X) = \underbrace{\alpha \|X\|_*}_{f(X)} + \underbrace{\mathbb{I}_{[a,b]}(X)}_{g(X)} + \underbrace{\tfrac{1}{2}\|\mathcal{P}_\Omega(X) - \mathcal{P}_\Omega(M)\|_F^2}_{w(X)} \right\} \tag{5.2}$$

where $\|\cdot\|_F$ denotes the Frobenius norm, $\mathbb{I}_{[a,b]}(X) = 0$ if $a \le X_{ij} \le b$ for all (i,j) and $\mathbb{I}_{[a,b]}(X) = \infty$ otherwise, and $\alpha > 0$ is a weighting parameter such that larger values of α promote lower rank solutions [55] in problem (5.2).

We generate the low-rank matrix as $M = L_1 L_2^T$ where $\{L_1, L_2\} \subset \mathbb{R}^{100\times 5}$ with entries chosen i.i.d. from $\mathcal{N}(3,1)$. This ensures M has rank 5 (with probability one) and that each entry is positive with high probability (each test instance was verified to have positive entries). We sample sn^2 entries of M uniformly at random, with a sampling ratio $s = 0.4$, i.e., 40% of the matrix M is observed in M_{obs}. We choose

$$\begin{aligned} a &= \min\{[M_{\text{obs}}]_{ij} \,|\, (i,j) \in \Omega\} - \sigma/2, \\ b &= \max\{[M_{\text{obs}}]_{ij} \,|\, (i,j) \in \Omega\} + \sigma/2 \end{aligned} \tag{5.3}$$

where σ is the standard deviation of all entries of M_{obs}.

We compare two frameworks: Davis-Yin (DY) (see Algo. 2) and ADMM (see Algo. 1). For each of these two frameworks, we consider the same three variants discussed in the previous section: no acceleration, acceleration based on decaying damping, and acceleration based on constant damping. These six algorithms are denoted by DY, DY-decaying, DY-constant, ADMM, ADMM-decaying, and ADMM-constant. Problem (5.2) can be solved using these algorithms with the proximal operator $J_{\tau\partial\|\cdot\|_*}(X) = UD_\tau(\Sigma)V^T$, where $X = U\Sigma V^T$ is the singular value decomposition of X and $[D_\tau(\Sigma)]_{ii} = \max\{\Sigma_{ii} - \tau, 0\}$; see [55] for details. The proximal operator of g is just the projection $\left[J_{\lambda\partial\mathbb{I}_{[a,b]}}(X)\right]_{ij} = \max\{a, \min(X_{ij}, b)\}$. Finally, $\nabla w(X) = \mathcal{P}_\Omega(X - M)$. In terms of algorithm parameters, we choose a step size of $\lambda = 1$ (for all variants), $r = 3$ for decaying damping, and $r = 0.1$ for constant damping. To evaluate algorithm performance, we use the relative error measure

$$\|M_k - M\|_F / \|M\|_F \tag{5.4}$$

where M_k is the solution estimate obtained during the kth iteration. The stopping criteria for the optimization algorithms is $\|M_{k+1} - M_k\|_F / \|M_k\|_F \le 10^{-10}$, which was satisfied for every problem instance even though it is a relatively tight tolerance.

In Fig. 4 we report the mean and standard deviation (errorbars) across 10 randomly generated instances of problem (5.2) with $\alpha = 3.5$ for the above algorithm variants. All methods terminate successfully and recover a matrix with the correct rank of five and a final

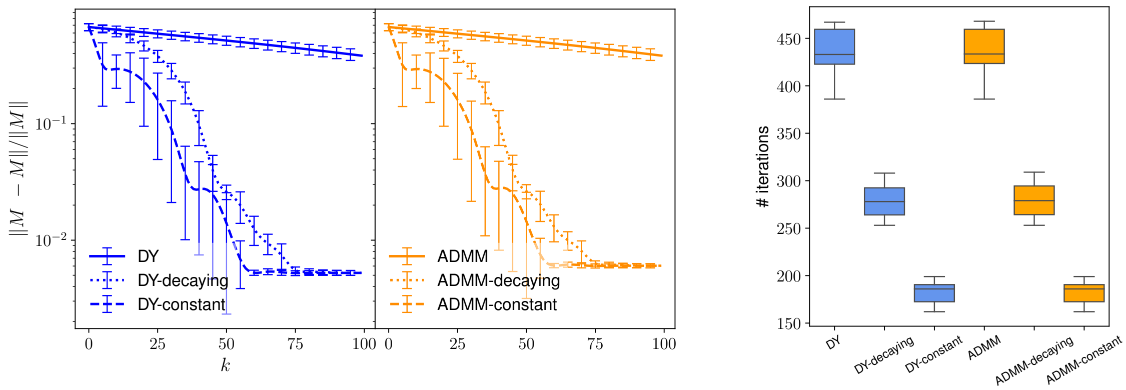

Figure 4: Performance of algorithms on problem (5.2) with $\alpha = 3.5$. We perform 10 Monte Carlo runs and indicate the mean and standard deviation for the relative error between the ground truth matrix M and the kth iterate M_k (left), and the number of iterations needed by the method to reach the termination tolerance (right).

relative error of $\approx 5 \times 10^{-3}$. The total number of iterations performed by each method are also shown in Fig. 4.

Motivated by the relatively large final relative error achieved for the single value of α in the previous paragraph, next we consider an annealing schedule on α that improves the relative error of the computed solutions. We follow the procedure of [56] as follows. Given a sequence $\alpha_1 > \alpha_2 > \cdots > \alpha_L = \bar{\alpha} > 0$ for some $\bar{\alpha}$, we run each algorithm with α_j and then use its solution as a starting point for the solution to the next run with α_{j+1}; all other parameters are kept fixed. Such an approach has been used in compressed sensing [57] and matrix completion [56]. Starting with $\alpha_0 = \delta\|M_{\text{obs}}\|_F$ for some $\delta \in (0, 1)$, we use the schedule $\alpha_{j+1} = \max\{\delta\alpha_j, \bar{\alpha}\}$ until reaching $\bar{\alpha}$. In our tests we choose $\delta = 0.25$ and $\bar{\alpha} = 10^{-8}$. We use the same algorithm parameters as those used in creating Fig. 4, except that for the constant damping variants we now use $r = 0.5$ since it performs better.

In Fig. 5 we report the mean and standard deviation (errorbars) across 10 randomly generated instances of problem (5.2). All methods successfully reach the termination tolerance, as for the previous test, but now achieve a much better reconstruction accuracy (compare Fig. 4 and Fig. 5). The total number of iterations for each method are also shown in Fig. 5. In this example the decaying damping variants do not improve over the non-accelerated method, but the constant damping variants still provide a speedup. We believe these findings can be explained by the fact that the accelerated gradient flow with constant damping attains exponential convergence on strongly convex problems, as opposed to the decaying damping (see Table 1).

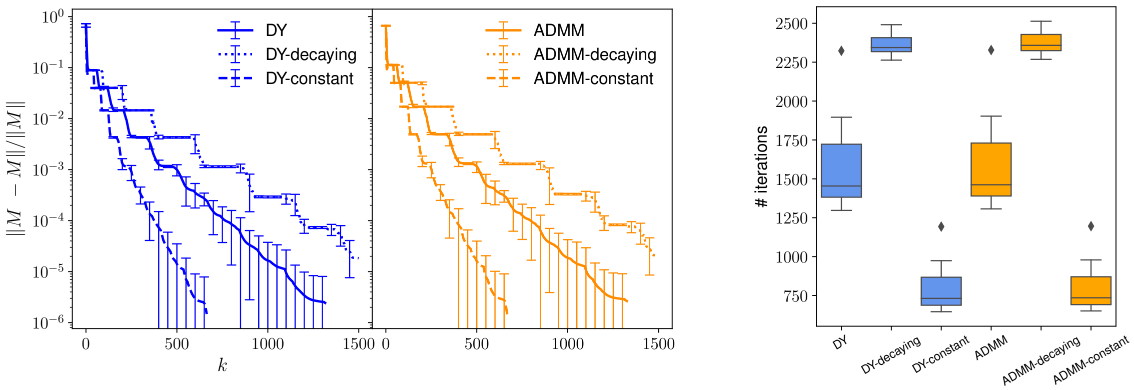

Figure 5: Performance of algorithms on problem (5.2) when annealing is used for α. We perform 10 Monte Carlo runs and indicate the mean and standard deviation for the relative error between the ground truth matrix M and the kth iterate M_k (left), and the number of iterations needed to reach the termination tolerance (right).

6 Final Remarks

We showed that four types of proximal algorithms, namely forward-backward, Tseng splitting, Douglas-Rachford, and Davis-Yin, correspond to different discretizations of the gradient flow (1.2). We also showed that several accelerated variants of each of these methods arise from a similar discretization to the accelerated gradient flow (1.3). Such algorithms are steady-state-preserving first-order integrators to the associated ODE. Moreover, we showed that ADMM and its accelerated variants correspond to a rebalanced splitting, which is a technique recently introduced in the literature [43] to obtain discretizations that preserve steady states.

The new accelerated frameworks (see Algos. 1, 2, and 3), which are new in general, reduce to known methods as special cases. Our frameworks provide different types of acceleration depending on the choice of damping strategy such as (1.8) or (3.12), although other choices are also possible.

Our derivations provide a new perspective on the important class of "operator splitting methods" by establishing tight connections with splitting methods for ODEs. Such an approach endows the gradient flow (1.2) and the accelerated gradient flow (1.3) with a unifying character for optimization since they capture the leading order behavior of several known algorithms. However, a complete understanding of a particular algorithm requires a more refined analysis and is an interesting problem.

Acknowledgments

We would like to thank Patrick Johnstone for discussions. This work was supported by grants ARO MURI W911NF-17-1-0304 and NSF 1447822.

References

[1] B. T. Polyak, “Some Methods of Speeding Up the Convergence of Iteration Methods,” *USSR Comp. Math. and Math. Phys.* **4** no. 5, (1964) 1–17.

[2] Y. Nesterov, “A Method of Solving a Convex Programming Problem with Convergence Rate $O(1/k^2)$,” *Soviet Mathematics Doklady* **27** no. 2, (1983) 372–376.

[3] W. Su, S. Boyd, and E. J. Candès, “A Differential Equation for Modeling Nesterov’s Accelerated Gradient Method: Theory and Insights,” *J. Mach. Learn. Res.* **17** no. 153, (2016) 1–43.

[4] A. Wibisono, A. C. Wilson, and M. I. Jordan, “A Variational Perspective on Accelerated Methods in Optimization,” *Proc. Nat. Acad. Sci.* **113** no. 47, (2016) E7351–E7358.

[5] W. Krichene, A. Bayen, and P. L. Bartlett, “Accelerated mirror descent in continuous and discrete time,” *NIPS* (2015) 2845–2853.

[6] H. Attouch, Z. Chbani, J. Peypouquet, and P. Redont, “Fast Convergence of Inertial Dynamics and Algorithms with Asymptotic Vanishing Viscosity,” *Math. Prog.* (2016) 1–53.

[7] A. C. Wilson, B. Recht, and M. I. Jordan, “A Lyapunov Analysis of Momentum Methods in Optimization.” arXiv:1611.02635v3 [math.OC], 2016.

[8] C. J. Maddison, D. Paulin, Y. W. Teh, B. O’Donoghue, and A. Doucet, “Hamiltonian Descent Methods.” arXiv:1809.05042 [math.OC], 2018.

[9] B. O’Donoghue and C. J. Maddison, “Hamiltonian Descent for Composite Objectives.” arXiv:1806.02608 [math.OC], 2019.

[10] M. Muehlebach and M. I. Jordan, “A Dynamical Systems Perspective on Nesterov Acceleration,” in *ICML*. 2019.

[11] G. França, J. Sulam, D. P. Robinson, and R. Vidal, “Conformal Symplectic and Relativistic Optimization.” arXiv:1903.04100 [math.OC], 2019.

[12] H. Attouch and A. Cabot, “Convergence of Damped Inertial Dynamics Governed by Regularized Maximally Monotone Operators,” *J. Diff. Eq.* **264** (2018) 7138–7182.

[13] H. Attouch and A. Cabot, “Convergence Rates of Inertial Forward-Backward Algorithms,” *SIAM J. Optim.* **28** no. 1, (2018) 849–874.

[14] R. May, “Asymptotic for a Second-Order Evolution Equation with Convex Potential and Vanishing Damping Term,” *Turkish J. Math.* **41** (2016) 681–785.

[15] H. Attouch, A. Cabot, Z. Chbani, and H. Riahi, “Inertial Forward-Backward Algorithms with Perturbations: Application to Tikhonov Regularization,” *J. Opt. Theo. and App.* **179** no. 1, (2018) 1–36.

[16] H. Attouch, J. Peypouquet, and P. Redont, “BackwardForward Algorithms for Structured Monotone Inclusions in Hilbert Spaces,” *J. Math. Anal. and App.* **457** no. 2, (2018) 1095–1117.

[17] G. França, D. P. Robinson, and R. Vidal, “ADMM and Accelerated ADMM as Continuous Dynamical Systems,” in *ICML*. 2018. arXiv:1805.06579 [math.OC].

[18] G. França, D. P. Robinson, and R. Vidal, “A Nonsmooth Dynamical Systems Perspective on Accelerated Extensions of ADMM.” arXiv:1808.04048 [math.OC], 2018.

[19] P. L. Lions and B. Mercier, “Splitting Algorithms for the Sum of two Nonlinear Operators,” *SIAM J. Numer. Anal.* **16** no. 6, (1979) 964–979.

[20] G. B. Passty, “Ergodic Convergence to a Zero of the Sum of Monotone Operators in Hilbert space,” *J. Math. Anal. Appl.* **72** no. 2, (1979) 383–390.

[21] S. P. Han and G. Lou, “A Parallel Algorithm for a Class of Convex Programs,” *SIAM J. Control Optim.* **29** (1991) 403–419.

[22] P. Tseng, “A Modified Forward-Backward Splitting Method for Maximal Monotone Mappings,” *SIAM J. Control Optim.* **38** no. 2, (2000) 431–446.

[23] J. Douglas and H. H. Rachford, “On the Numerical Solution of Heat Conduction Problems in two and Three Space Variables,” *Trans. Amer. Math. Soc.* **82** (1956) 421–439.

[24] D. Davis and W. Yin, “A Three-Operator Splitting Scheme and its Optimization Applications,” *Set-Valued and Variational Analysis* **25** (2017) 829–858.

[25] R. Glowinski and A. Marroco, “Sur l’approximation, par él’ements finis d’ordre un, et la résolution, par pénalisation-dualité d’une classe de probèmes de Dirichlet non linéaires,” *ESAIM: Mathematical Modelling and Numerical Analysis - Modélisation Mathématique et Analyse Numérique* **9** no. R2, (1975) 41–76.

[26] D. Gabay and B. Mercier, “A Dual Algorithm for the Solution of Nonlinear Variational Problems via Finite Element Approximations,” *Computers and Mathematics with Applications* **2** no. 1, (1976) 17–40.

[27] P. R. Johnstone and J. Eckstein, “Projective Splitting with Forward Steps: Asynchronous and Block-Iterative Operator Splitting.” arXiv:1803.07043 [math.OC], 2018.

[28] 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* **3** no. 1, (2011) 1–122.

[29] F. E. Browder, “Nonlinear Elliptic Boundary Value Problems,” *Bull. Amer. Math. Soc.* **69** no. 6, (1963) 862–874.

[30] F. E. Browder, “The Solvability of Non-linear Functional Equations,” *Duke Math J.* **30** no. 4, (1963) 557–566.

[31] F. E. Browder, “Variational Boundary Value Problems for Quasi-Linear Elliptic Equations of Arbitrary Order,” *Proc. Nat. Acad. Sci.* **50** no. 1, (1963) 31–37.

[32] R. T. Rockafellar, “Monotone Operators and the Proximal Point Algorithm,” *SIAM J. Control Optim.* **14** no. 5, (1976) 877–898.

[33] P. L. Combettes, “Solving Monotone Inclusions via Compositions of Nonexpansive Averaged Operators,” *Optimization* **53** no. 5, (2004) 475–504.

[34] P. L. Combettes and J. C. Pesquet, “Proximal Splitting Methods in Signal Processing,” *Fixed-Point Algorithms for Inverse Problems in Science and Engineering* (2011) 185–212.

[35] P. L. Combettes and V. R. Wajs, “Signal Recovery by Proximal Forward-Backward Splitting,” *Multiscale Model. Simul.* **4** no. 4, (2005) 1168–1200.

[36] O. Güler, “On the Convergence of the Proximal Point Algorithm for Convex Minimization,” *SIAM J. Control Optim.* **29** (1991) 403–419.

[37] J. Eckstein and D. P. Bertsekas, “On the Douglas-Rachford Splitting Method and the Proximal Point Algorithm for Maximal Monotone Operators,” *Mathematical Programming* **55** (1992) 293–318.

[38] H. H. Bauschke and P. L. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces.* Springer International Publishing, 2017.

[39] E. K. Ryu and S. Boyd, “A Primer on Monotone Operator Methods,” *Appl. Comput. Math.* **15** no. 1, (2016) 3–43.

[40] A. Cauchy, “Méthode générale pour la résolution des systèmes d’équations simultanées,” *C. R. Acad. Sci. Paris* **25** (1847) 536–538.

[41] R. I. McLachlan and G. R. W. Quispel, “Splitting Methods,” *Acta Numerica* **11** (2002) 341–434.

[42] S. MacNamara and G. Strang, *Operator Splitting*, pp. 95–114. Springer International Publishing, 2016.

[43] R. L. Speth, W. H. Green, S. MacNamara, and G. Strang, “Balanced Splitting and Rebalanced Splitting,” *SIAM J. Numer. Anal.* **51** no. 6, (2013) 3084–3105.

[44] B. Martinet, “Regularisation dinéquations variationelles par approximations successives,” *Rev. Française Informat. Recherche Opérationnelle* **4** (1970) 154–158.

[45] B. Martinet, “Determination approchée dun point fixe dune application pseudo-contractante,” *C. R. Acad. Sci. Paris Ser. A-B* **274** (1972) 163–165.

[46] R. T. Rockafellar, “Monotone Operators and the Proximal Point Algorithm,” *SIAM J. Control Optim.* **14** (1976) 877–898.

[47] R. T. Rockafellar, “Augmented Lagrangians and Applications of the Proximal Point Algorithm in Convex Programming,” *Math. Oper. Res.* **1** (1976) 97–116.

[48] E. Hairer, C. Lubich, and G. Wanner, *Geometric Numerical Integration.* Springer, 2006.

[49] S. Blanes, F. Casas, and A. Murua, “Splitting and Composition Methods in the Numerical Integration of Differential Equations,” *Bol. Soc. Esp. Mat. Apl.* **45** (2008) 89–145.

[50] J. C. Butcher, *Numerical Methods for Ordinary Differential Equations.* John Wiley & Sons, 2008.

[51] T. Goldstein, B. O’Donoghue, S. Setzer, and R. Baraniuk, “Fast Alternating Direction Optimization Methods,” *SIAM J. Imag. Sci.* **7** no. 3, (2014) 1588–1623.

[52] H. H. Bauschke and W. M. Moursi, “On the Douglas-Rachford Algorithm,” *Math. Program.* **164** (2017) 263–284.

[53] P. Patrinos, L. Stella, and A. Bemporad, “Douglas-Rachford Splitting: Complexity Estimates and Accelerated Variants,” in *53rd IEEE Conf. Dec. and Control*, pp. 4234–4239. 2014.

[54] J.-P. Aubin and A. Cellina, *Differential Inclusions.* Springer-Verlag, 1984.

[55] J. Cai, E. Candès, and Z. Shen, “A Singular Value Thresholding Algorithm for Matrix Completion,” *SIAM Journal on Optimization* **20** no. 4, (2010) 1956–1982.

[56] S. Ma, D. Goldfarb, and L. Chen, “Fixed Point and Bregman Iterative Methods for Matrix Rank Minimization,” *Math. Prog.* **128** (2011) 321–353.

[57] E. T. Hale, W. Yin, and Y. Zhang, “Fixed-Point Continuation Applied to Compressed Sensing: Implementation and Numerical Experiments,” *J. Comp. Math.* **28** no. 2, (2010) 170–194.