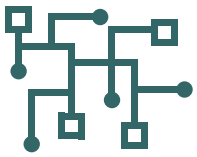

ISE

Industrial and Systems Engineering

Gradient Flows and Accelerated Proximal Splitting Methods

Guilherme Franca[1], Daniel P. Robinson[2], and Rene Vidal[1]

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

[2]Industrial and Systems Engineering, Lehigh University

ISE Technical Report 19T-020

LEHIGH UNIVERSITY

Gradient Flows and Accelerated Proximal Splitting Methods

Guilherme França*, Daniel P. Robinson, René Vidal

Mathematical Institute for Data Science,
Johns Hopkins University, Baltimore MD 21218, USA
and
Industrial and Systems Engineering,
Lehigh University, Bethlehem, PA 18015, USA

Abstract

Proximal based algorithms are well-suited to nonsmooth optimization problems with important applications in signal processing, control theory, statistics and machine learning. There are essentially four basic types of proximal algorithms based on fixed-point iteration currently known: forward-backward splitting, forward-backward-forward or Tseng splitting, Douglas-Rachford, and the very recent 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 these different methods can be derived from the gradient flow by using splitting methods for ordinary differential equations. Furthermore, applying similar discretization scheme to a particular second order differential equation results in accelerated variants of the respective algorithm, which can be of Nesterov or heavy ball type, although we treat both simultaneously. Many of the optimization algorithms we derive are new. For instance, we propose accelerated variants of Davis-Yin and two extensions of ADMM together with their accelerated variants. Interestingly, we show that (accelerated) ADMM corresponds to a rebalanced splitting which is a recent technique designed to preserve steady states of the differential equation. Overall, our results strengthen the connections between optimization and continuous dynamical systems and offer a more unified perspective on accelerated methods.

*guifranca@gmail.com

Contents

1 Introduction

There has been an increasing interest in establishing connections between optimization and continuous dynamical systems, especially for accelerated gradient based methods [1–9]. More recently, extensions of such connections to nonsmooth cases using proximal methods started to be considered as well [10–16]. Proximal algorithms play an important role in modern science since they enjoy an improved stability, can be applied under very mild assumptions, and in many cases the associated proximal operators admit simple closed form expressions leading to reasonably cheap and efficient algorithms. On a fundamental level, the majority of known proximal methods can traced back to the following types:

- forward-backward splitting [17–19];
- forward-backward-forward or Tseng splitting [20];
- Douglas-Rachford [17, 21];

- Davis-Yin three-operator splitting [22];
- and closely related, although of a different class, we also have the alternating direction method of multipliers (ADMM) [23, 24].

Many other more sophisticated optimization methods consist of extensions or variations of these basic types. The first three were the only known methods for a very long time. Only recently Davis-Yin solved the long term problem of obtaining a three-operator splitting that cannot be reduced to any of the previous existing two-operator splitting schemes [22]. These methods are based on finding fixed points of a nonexpansive monotone operator. A different technique, that we will not consider in this paper, based on projection onto separating sets has also been recently proposed; see [25] and references therein. ADMM dates back to the 70's but only in the last decade has been gaining increasing popularity [26] thanks to its effectiveness in solving problems with sparse and low rank regularization, common in regression, dictionary learning, matrix completion, and robust principal component analysis, to name a few.

In this paper, we show that all of the above mentioned proximal optimization methods can actually be derived as discretizations of the very same continuous dynamical system, namely the simple *gradient flow*. We show this through connections with *splitting methods* for ordinary differential equations (ODEs) which is a powerful technique for algorithm building in numerical analysis [27, 28]. Interestingly, we show that ADMM corresponds to a *rebalanced splitting* which is a recent technique designed to preserve steady states of the underlying ODE [29]. Furthermore, we show that accelerated variants of all these algorithms arise as different splittings of the same second order ODE, namelly the *accelerated gradient flow*. To the best of our knowledge, most of the accelerated algorithms we introduce are new in the literature, although some recover known algorithms as particular cases. Thus, our results establish tight relationships between optimization and elegant techniques in numerical analysis of ODEs.

1.1 Some Basic Building Blocks

Let us provide a bit of context to motivate our approach. The first connection between optimization and a continuous dynamical system dates back to Cauchy [30] who considered the *gradient flow*

$$\dot{x} = -\nabla f(x), \tag{1}$$

where $x = x(t) \in \mathbb{R}^n$, $f : \mathbb{R}^n \to \mathbb{R}$ and $\dot{x} \equiv \frac{dx}{dt}$. Under an explicit Euler discretization of this ODE one has

$$\frac{x_{k+1} - x_k}{\lambda} = -\nabla f(x_k), \tag{2}$$

where $x_k \equiv x(t_k)$ with $t_k = k\lambda$, for $k = 0, 1, \ldots$, and $\lambda > 0$ is the stepsize. One immediately recognizes (2) as the *gradient descent* method. Alternatively, an implicit Euler discretization of this same ODE yields

$$\frac{x_{k+1} - x_k}{\lambda} = -\nabla f(x_{k+1}), \tag{3}$$

which now requires solving a nonlinear equation in x_{k+1}. Fortunately, this can be done through the *resolvent operator* defined as

$$J_{zA} \equiv (I + zA)^{-1} \tag{4}$$

where A is an operator and $z \in \mathbb{C}$ is the spectral parameter.[1] Herein, we are interested in cases where A is a maximal monotone operator and we always choose $z \equiv \lambda > 0$ to be a real number related to the discretization stepsize. For instance, when $A = \nabla f$ the resolvent is equal to the *proximal operator*[2]

$$J_{\lambda\nabla f}(v) \equiv \operatorname{prox}_{\lambda f}(v) \equiv \arg\min_x \left(f(x) + \tfrac{1}{2\lambda}\|x - v\|^2 \right). \tag{5}$$

Hence, we obtain from (3) the *proximal point algorithm*

$$x_{k+1} = J_{\lambda\nabla f}(x_k). \tag{6}$$

This method was proposed by [32, 33] and later extensively studied and generalized [34–36]. For a convex f it is known to converge under the mere assumption that a minimizer exists. The proximal point method is more stable compared to gradient descent and allows a much more agressive choice of stepsize, as common in implicit discretizations. However, it has the extra cost of computing proximal operators instead of gradients. Importantly, the proximal point method can be employed even when f is non differentiable, as opposed to gradient descent. These two examples, namely (2) and (6), partially illustrate one of the ideas that we wish to explore in more detail: *different optimization algorithms emerge from different discretizations of the same continuous dynamical system.* This perspective, if true, gives a central character to the continuous dynamical system whose behaviour becomes a valuable guiding principle for optimization.

As it turns out, the idea of connecting continuous dynamical systems and optimization has also been explored to understand properties of *accelerated methods*. Such algorithms are

[1] The resolvent was introduced by Fredholm in the late 19th century to study integral equations related to partial differential equations. Later it was introduced by Hilbert in the theory of linear operators. Usually, the resolvent is defined as $R(z) \equiv (A - zI)^{-1}$ which can be used to study the spectral decomposition of A. In convex analysis the resolvent is usually defined as in (4) which is obviously related as $J_{\lambda A} = \lambda^{-1} R(-\lambda^{-1})$.

[2] This holds for non differentiable functions as well where $A = \partial f$ is the subdifferential of f. In this case the differential equation (1) is replaced by a differential inclusion. (See Section 4 and [31] for details on resolvent formalism for general montone operators.) Recall that $\partial f(x) = \{\nabla f(x)\}$ when f is differentiable.

considered counterintuitive mostly due to their heuristic construction since introduced by Nesterov [37]. Another accelerated method was introduced even before that by Polyak [38] and is called the heavy ball method. In general, it is unclear how to "accelerate" a given algorithm. As we will show, an accelerated variant of a specific algorithm, that is obtained as a particular discretization of (1), can be obtained by a similar discretization strategy but applied to the second order differential equation

$$\ddot{x} + \eta(t)\dot{x} = -\nabla f(x) \tag{7}$$

which henceforth we call the *accelerated gradient flow*. Above,

$$\eta(t) \equiv \begin{cases} r/t & \text{for Nesterov } (r \geq 3), \quad (8a) \\ r & \text{for heavy ball } (r > 0). \quad (8b) \end{cases}$$

It has recently been shown [1] that the continuous limit of Nesterov's accelerated gradient [37] corresponds to (7) with damping coefficient (8a). It is also known that Polyak's heavy ball method [38] is a discretization of (7) with (8b). Both methods are obtained by explicit Euler discretization and correspond to accelerated variants of gradient descent. Therefore, at least in this case, it is clear that applying similar discretization technique to both (1) and (7) yield an algorithm together with its accelerated variant. We will extend this perspective to the basic types of proximal optimization algorithms currently known.

We focus on proximal algorithms that were previously introduced from an operator splitting approach in convex analysis. The literature on operator splitting is huge and unfortunately we cannot make due justice to all important papers. We just mention that these methods have origins in functional analysis and differential equations [39–41] and later started to be explored in convex analysis and optimization; see [36, 42–46] and references therein for a survey. We also refer to [31, 47] for introduction and historical account.

Since both dynamical systems (1) and (7) play the central role in our analysis, for further reference we summarize some of their convergence rates in Table 1. Note that these rates hold in continuous time. One expects that a reasonable discretization would preserve the same rates, at least up to a small error, however a discrete analysis is necessary to formally establish them, which often is a challenging task. Nevertheless, knowing the continuous rates is valuable especially when analogous result is unknown in discrete time. The scope of this paper is not on establishing discrete rates but rather on expanding the connections between optimization and ODEs.

1.2 Outline

We organize our paper as follows. In Section 2, we introduce the so-called balanced and rebalanced splitting approaches proposed in [29]. For the rebalanced case we propose a

	convex $(f(x(t)) - f^\star)$	strongly convex $(\|x(t) - x^\star\|)$
gradient flow	$\mathcal{O}\left(t^{-1}\right)$	$\mathcal{O}\left(e^{-mt/2}\right)$
Nesterov	$\mathcal{O}\left((r-1)^2 t^{-2}\right)$	$\mathcal{O}\left(r^{r/3} m^{-(3+r)/6} t^{-r/3}\right)$
heavy ball	$\mathcal{O}\left(rt^{-1}\right)$	$\mathcal{O}\left(rm^{-1/2} e^{-rt/3}\right)$ with $r \leq \frac{3}{2}\sqrt{m}$

Table 1: Convergence rates of the gradient flow (1) and accelerated gradient flow (7). Let $f^\star \equiv \min f$, and for an m-strongly convex function let $x^\star$ be the unique minimizer. We show rates in terms of $f(x(t)) - f^\star$ for convex functions, while $\|x(t) - x^\star\|$ for strongly convex functions. Note the interesting tradeoff between Nesterov and heavy ball acceleration for convex versus strongly convex settings. The above rates were proven in [16].

slight modification compared to [29] that will enable us to make connections with (known) optimization algorithms. In Section 3, we derive the Davis-Yin, ADMM, and Tseng splitting methods from an ODE splitting approach. The results for Davis-Yin automatically imply the same connections to forward-backward splitting and Douglas-Rachford as particular cases. More importantly, we obtain several accelerated variants of these algorithms, most of which seem to be new. We also show that all these discretizations correspond to a first order integrator of the associated ODE. Moreover, we show how these different splittings preserve steady states of the original ODE. In Section 4, we argue that all the previous analysis can be extended to nonsmooth cases, and in more general to maximal monotone operators. Finally, in Section 5 we show some numerical experiments illustrating the speedup achieved by our proposed methods.

2 Splitting Methods for ODEs

In this section we introduce the basic ideas about splitting methods for ODEs that will be used throughout the paper.

In the words of MacNamara and Strang [28] *"... there are only ten big ideas in numerical analysis; all the rest are merely variations on those themes ..."*. Splitting is certainly one of them. To consider this idea in the context of ODEs, assume that

$$\dot{x} = \varphi(x) \tag{9}$$

consists of an intractable problem. We denote its flow map by Φ_t. The idea is to split the vector field $\varphi : \mathbb{R}^n \to \mathbb{R}^n$ into separable parts, each integrable or amenable to a feasible numerical approximation. For simplicity, let

$$\varphi = \varphi^{(1)} + \varphi^{(2)} \tag{10}$$

and suppose that both subproblems

$$\dot{x} = \varphi^{(1)}(x), \qquad \dot{x} = \varphi^{(2)}(x), \tag{11}$$

are completely feasible, yielding the respective flow maps $\Phi_t^{(1)}$ and $\Phi_t^{(2)}$. Let h be a fixed stepsize. It can then be shown that the simplest composition [48]

$$\hat{\Phi}_h = \Phi_h^{(2)} \circ \Phi_h^{(1)} \tag{12}$$

already provides a first order approximationto the true flow, i.e. $\|\Phi_h(x) - \hat{\Phi}_h(x)\| = \mathcal{O}(h^2)$. This immediately implies that $\|\Phi_t(x) - (\hat{\Phi}_h)^k(x)\| = \mathcal{O}(h)$ for a finite interval $t = hk$.

There are several ways one can compose individual flows, each resulting in a different method. For instance, the composition $\hat{\Phi}_h = \Phi_{h/2}^{(1)} \circ \Phi_h^{(2)} \circ \Phi_{h/2}^{(1)}$ is a method of order $p = 2$ [49]. Given a base method $\hat{\Phi}_h$ of even order p one can also form more elaborate compositions. For example, $\Psi_h = \hat{\Phi}_{\gamma_s h} \circ \hat{\Phi}_{\gamma_{s-1} h} \circ \cdots \circ \hat{\Phi}_{\gamma_1 h}$ with real coefficients γ_i such that $\gamma_1 + \cdots + \gamma_s = 1$ and $\gamma_1^{p+1} + \cdots + \gamma_s^{p+1} = 0$ yields a method of order at least $p+1$. One can also use a preprocessor map $\chi_h : \mathbb{R}^n \to \mathbb{R}^n$ such that $\tilde{\Phi}_h = \chi_h^{-1} \circ \hat{\Phi}_h \circ \chi_h$ is more accurate than $\hat{\Phi}_h$ and basically introduces no extra cost. There are many interesting ideas in splitting methods for ODEs, some quite sophisticated. We simply mention these facts to give the reader a feeling that there is much more than will be explored in this paper. We refer to [27, 48, 50] for these and many other interesting details. Naturally, more accurate methods are more expensive since they involve extra computations 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 (12) that suffices to make connections with optimization methods.

2.1 Balanced Splitting

In general, splitting methods such as (12) do not preserve steady states of the ODE (9). Recently, an approach designed to preserve steady states was proposed [29] under the name of *balanced splitting*. The idea is to introduce a balance coefficient c in the form

$$\dot{x} = \varphi^{(1)}(x) + c, \qquad \dot{x} = \varphi^{(2)}(x) - c, \tag{13}$$

such that asymptotically

$$c_\infty = \tfrac{1}{2}\big(\varphi^{(2)}(x_\infty) - \varphi^{(1)}(x_\infty)\big). \tag{14}$$

A stationary state x_∞ of (9) is by definition such that $\big(\varphi^{(1)} + \varphi^{(2)}\big)(x_\infty) = 0$. Therefore, each equation in (13) obeys

$$\dot{x}_\infty = \varphi^{(1)}(x_\infty) + c_\infty = \tfrac{1}{2}\big(\varphi^{(1)} + \varphi^{(2)}\big)(x_\infty) = 0, \tag{15a}$$

$$\dot{x}_\infty = \varphi^{(2)}(x_\infty) - c_\infty = \tfrac{1}{2}\big(\varphi^{(2)} + \varphi^{(1)}\big)(x_\infty) = 0, \tag{15b}$$

showing that stationary states are preserved in both equations. In practice, this is implemented by computing the quantity $c_k = \frac{1}{2}\big(\varphi^{(2)}(x_k) - \varphi^{(1)}(x_k)\big)$ at each step of the method, followed by appropriate discretizations of the individual equations in (13). Note that this approach requires explicit computations of $\varphi^{(i)}$ which may not be available in an optimization problem where $\varphi^{(i)} = -\nabla f^{(i)}$ is not defined for a nonsmooth function $f^{(i)}$ entering the composite objective function $f = f^{(1)} + f^{(2)}$.

2.2 Rebalanced Splitting

The *rebalanced splitting* approach was also proposed by [29], which is more stable than the balanced splitting previously discussed and also allows one to compute the balance coefficient using only the previous iterates. We now introduce a slight variant of this method. Let $t_k = kh$, for $k = 0, 1, \ldots$, and integrate $\dot{x} = \varphi^{(1)}(x) + c$ with initial condition $x(t_k) = x_k$ up to the final point $x_{k+1/2}$, and also $\dot{x} = \varphi^{(2)}(x) - c$ with initial condition $x(t_k) = x_{k+1/2}$. The integration interval is $[t_k, t_k + h]$. We thus obtain the two updates[3]

$$x_{k+1/2} = x_k + \int_{t_k}^{t_k+h} \big(\varphi^{(1)} + c_k\big)dt, \tag{16a}$$

$$x_{k+1} = x_{k+1/2} + \int_{t_k}^{t_k+h} \big(\varphi^{(2)} - c_k\big)dt. \tag{16b}$$

We now use the *average* of $\varphi^{(1)}$ and $\varphi^{(2)}$ over such timestep to obtain (14) through

$$\begin{aligned} c_{k+1} &= \frac{1}{2h}\int_{t_k}^{t_k+h} \varphi^{(2)}dt - \frac{1}{2h}\int_{t_k}^{t_k+h} \varphi^{(1)}dt \\ &= c_k + \frac{1}{h}\left(\frac{x_{k+1} + x_k}{2} - x_{k+1/2}\right). \end{aligned} \tag{17}$$

Note that we used (16) to replace the integrals. Contrary to the previous balanced case, such an approach does not require explicitly computing $\varphi^{(i)}$ to obtain c_{k+1} since it is computed from the previous iterates. Therefore, this procedure is well-suited to nonsmooth optimization since it will not require explicit gradient computations (this point will be more clear shortly when we use implicit discretizations).

3 Proximal Algorithms from ODE Splitting

We now use the ideas introduced in the previous section to construct implicit discretizations of both the gradient flow (1) and the accelerated gradient flow (7). For simplicity, in the

[3] We denote the initial and final states related by the overall map as $x_{k+1} = \hat{\Phi}_h(x_k)$ and intermediate states such as obtained from $\Phi_h^{(1)}$ by "fractional indices" such as $x_{k+1/2}$.

following we assume that all functions are differentiable. This is by no means necessary and later in Section 4 we explain how this assumption can be droped. We just do not want to overwhelm the analysis and notation by introducing differential inclusions or regularized monotone operators at this stage.

Consider the gradient flow under a composite function $f + g + w$, i.e.

$$\dot{x} = -\nabla f(x) - \nabla g(x) - \nabla w(x) \tag{18}$$

where $f, g, w : \mathbb{R}^n \to \mathbb{R}$. Consider also the composite accelerated gradient flow

$$\ddot{x} + \eta(t)\dot{x} = -\nabla f(x) - \nabla g(x) - \nabla w(x). \tag{19}$$

The damping coefficient η is given by (8). We will treat both Nesterov and heavy ball type of acceleration in a unified manner. One has the standard discretizations

$$\dot{x}(t_k) = (x_{k+1} - x_k)/h + \mathcal{O}(h), \tag{20}$$

$$\ddot{x}(t_k) = (x_{k+1} - 2x_k + x_{k-1})/h^2 + \mathcal{O}(h), \tag{21}$$

where $t_k = kh$, for $k = 0, 1, \ldots$, $x_k \equiv x(t_k)$ and $h > 0$ is the stepsize. For the differential operator in (19) let us define

$$\hat{x}_k \equiv x_k + \gamma_k(x_k - x_{k-1}) \tag{22}$$

with

$$\gamma_k \equiv \begin{cases} k/(k+r) & \text{for Nesterov } (r \geq 3), \quad (23a) \\ 1 - rh & \text{for heavy ball } (r > 0). \quad (23b) \end{cases}$$

One can readily check that

$$\ddot{x}(t_k) + \eta(t_k)\dot{x}(t_k) = (x_{k+1} - \hat{x}_k)/h^2 + \mathcal{O}(h). \tag{24}$$

As usual, we will keep the leading order and neglect $\mathcal{O}(h)$ terms altogether. Thus, it is straightforward to conclude the following.

Lemma 1. *An implicit Euler discretization of the gradient flow* (1) *yields*

$$x_{k+1} = J_{\lambda\nabla f}(x_k), \tag{25}$$

where $h = \lambda$ is the stepsize. Analogously, an implicit discretization of the accelerated gradient flow (7)*, with stepsize $h^2 = \lambda$, yields*

$$x_{k+1} = J_{\lambda\nabla f}(\hat{x}_k), \tag{26}$$

where $\hat{x}_k$ is defined in (23) *and accounts for both Nesterov and heavy ball acceleration.*

Note that (25) is the proximal point algorithm (6), while (26) corresponds to its respective Nesterov or heavy ball accelerated variant according with the choice of γ_k in (22).

Lemma 1 will form the basis to construct other more elaborate splitting schemes to the ODEs (18) and (19). Note that we choose the stepsize to be related to the spectral parameter of the resolvent (4). For the gradient flow we choose $h = \lambda$, while for the accelerated gradient flow we choose $h = \sqrt{\lambda}$. Hence, for fixed k and λ, the time elapsed for the former is $t_k = k\lambda$, while for the latter we have $t_k = k\sqrt{\lambda}$ which is larger, highlighting the acceleration of a second order ODE compared to a first order one.

To make connections between our ODE splitting schemes and operator splitting schemes commonly used in optimization we will make use of the following result.

Lemma 2. *It holds that*

$$(\nabla f + \nabla g + \nabla w)(\bar{x}) = 0 \iff \mathcal{P}(x) = x \tag{27}$$

where

$$\mathcal{P} \equiv \tfrac{1}{2}I + \tfrac{1}{2}C_{\lambda\nabla g} \circ (C_{\lambda\nabla f} - \lambda\nabla w \circ J_{\lambda\nabla f}) - \tfrac{1}{2}\lambda\nabla w \circ J_{\lambda\nabla f}, \tag{28}$$

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

Proof. The leftmost equality in (27) is equivalent to $(I + \lambda\nabla g)(\bar{x}) = (I - \lambda\nabla f - \lambda\nabla w)(\bar{x})$. Using the resolvent (4) this is equivalent to

$$\bar{x} = J_{\lambda\nabla g} \circ (I - \lambda\nabla f - \lambda\nabla w)\,(\bar{x}). \tag{29}$$

Using the identity

$$C_{\lambda\nabla f} \circ (I + \lambda\nabla f) = \big(2(I + \lambda\nabla f)^{-1} - I\big) \circ (I + \lambda\nabla f) = I - \lambda\nabla f \tag{30}$$

and replacing $J_{\lambda\nabla g} = \frac{1}{2}(C_{\lambda\nabla g} + I)$ into (29) we have

$$\bar{x} = \tfrac{1}{2}\,(C_{\lambda\nabla g} + I) \circ \{C_{\lambda\nabla f} \circ (I + \lambda\nabla f) - \lambda\nabla w\}\,(\bar{x}). \tag{31}$$

Let $x \equiv (I + \lambda\nabla f)(\bar{x})$, or equivalently $\bar{x} = J_{\lambda\nabla f}(x)$. Hence,

$$J_{\lambda\nabla f}(x) = \tfrac{1}{2}C_{\lambda\nabla g} \circ (C_{\lambda\nabla f} - \lambda\nabla w \circ J_{\lambda\nabla f})\,(x) + \tfrac{1}{2}C_{\lambda\nabla f}(x) - \tfrac{1}{2}\lambda\nabla w \circ J_{\lambda\nabla f}(x) \tag{32}$$

which is equivalent to

$$\tfrac{1}{2}x = \tfrac{1}{2}C_{\lambda\nabla g} \circ (C_{\lambda\nabla f} - \lambda\nabla w \circ J_{\lambda\nabla f})\,(x) - \tfrac{1}{2}\lambda\nabla w \circ J_{\lambda\nabla f}(x). \tag{33}$$

Adding $x/2$ on each side yields the rightmost identity of (27) with (28), as claimed.[4] □

[4] It is possible to generalize this result to general maximal monotone operators, i.e. $0 \in (A+B+C)\bar{x} \iff \mathcal{P}x = x$, where $\mathcal{P} \equiv \frac{1}{2}I + \frac{1}{2}C_{\lambda B} \circ (C_{\lambda A} - \lambda C \circ J_{\lambda A}) - \frac{1}{2}\lambda C \circ J_{\lambda A}$, $\bar{x} = J_{\lambda A}x$ and C must be single-valued. One just needs to be careful with appropriate inclusions and note that the resolvent is single-valued.

Algorithm 1 Davis-Yin algorithm [22] for solving (34). This algorithm reduces to Douglas-Rachford when $w = 0$ and to forward-backward when $f = 0$.

for $k = 1, 2, \ldots$ **do**
  $x_{k+1/4} \leftarrow J_{\lambda\nabla f}(x_k)$
  $x_{k+1/2} \leftarrow 2x_{k+1/4} - x_k$
  $x_{k+3/4} \leftarrow J_{\lambda\nabla g}\big(x_{k+1/2} - \lambda\nabla w(x_{k+1/4})\big)$
  $x_{k+1} \leftarrow x_k + x_{k+3/4} - x_{k+1/4}$
end for

Fxed points of (28) are zeros of $\nabla f + \nabla g + \nabla w$, i.e. stationary points of the function $f + g + w$. Building iterations that reproduce fixed points of an operator such as (28) is how one builds algorithms from an operator splitting approach [22, 47]. In the past decade there has been increasing interest in operator splitting methods for signal processing, statistics and machine learning, since they are simple and usually well-suited to large scale problems.

3.1 Davis-Yin Three-Operator Splitting

The Davis-Yin method [22] solves the optimization problem

$$\min_{x\in\mathbb{R}^n} f(x) + g(x) + w(x) \tag{34}$$

through the updates shown in Algorithm 1. It was shown that for convex functions this method has a convergence $\mathcal{O}(1/k)$ in an average or ergodic sense. Moreover, when all functions are strongly convex and under some regularity conditions, linear (or exponential) convergence holds (see [22]). Note that when $w = 0$ this algorithm is equivalent to Douglas-Rachford [17, 21], while when $f = 0$, so that $J_{\lambda\nabla f} = I$, this method reduces to the forward-backward splitting given by the single update $x_{k+1} = J_{\lambda\nabla g}(x_k - \lambda\nabla w(x_k))$ [18].

To see how Algorithm 1 arises from an ODE splitting approach, consider a *two-splitting* of the composite gradient flow (18) given by

$$\dot{x} = -\nabla f(x), \qquad \dot{x} = -\nabla g(x) - \nabla w(x). \tag{35}$$

According to Lemma 1, an implicit discretization of the first equation results into

$$x_{k+1/4} = \Phi_h^{(1)}(x_k) = J_{\lambda\nabla f}(x_k) \tag{36}$$

where $\Phi_h^{(1)}$ denotes the numerical flow and $h = \lambda$ is the stepsize. Now we introduce a transformation $\mathcal{T}_h$ that translates this vector a distance twice apart, i.e. we require that the next point $x_{k+1/2}$ obeys $x_{k+1/2} - x_k = 2(x_{k+1/4} - x_k)$, or equivalently

$$x_{k+1/2} = \mathcal{T}_h(x_{k+1/4}) = 2x_{k+1/4} - x_k. \tag{37}$$

Note that this is just the Caley operator (see Lemma 2) acting on the starting point x_k, i.e. $\mathcal{T}_h(x_{k+1/4}) = C_{\lambda\nabla f}(x_k)$. We can also write

$$\mathcal{T}_h(x_{k+1/4}) = \left(2I - J^{-1}_{\lambda\nabla f}\right)(x_{k+1/4}) = (I - \lambda\nabla f)(x_{k+1/4}) \tag{38}$$

where we used the resolvent (4) and (36). Thus, $\mathcal{T}_h = I - \lambda\nabla f$ is well-defined, being just a gradient descent step. Moreover, this operator has an inverse $\mathcal{T}_h^{-1}$. Now, consider a semi-implicit discretization of the second equation in (35), i.e.

$$\frac{x_{k+3/4} - x_{k+1/2}}{\lambda} = -\nabla g(x_{k+3/4}) - \nabla w(x_{k+1/4}). \tag{39}$$

Using the resolvent (4) this is equivalent to

$$x_{k+3/4} = \Phi_h^{(2)}(x_{k+1/2}) = J_{\lambda\nabla g}\left(x_{k+1/2} - \lambda\nabla w(x_{k+1/4})\right). \tag{40}$$

Finally, we apply the inverse $\mathcal{T}_h^{-1}$ to discount the previous translation. Instead of computing this operator explicitly, note that we just need to subtract the vector $x_{k+1/2} - x_{k+1/4}$ which was used in obtaining (37). Hence,

$$\begin{aligned} x_{k+1} &= \mathcal{T}_h^{-1}(x_{k+3/4}) \\ &= x_{k+3/4} - (x_{k+1/2} - x_{k+1/4}) \\ &= x_{k+3/4} - (x_{k+1/4} - x_k). \end{aligned} \tag{41}$$

The sequence of steps (36), (37), (40) and (41) are precisely Algorithm 1. An illustration of this discretization scheme is shown in Fig. 1. Interestingly, this algorithm was proposed to solve the long term problem of obtaining a *three-operator splitting* that cannot be reduced to the existing two-operator splitings. We see that in the context of ODEs, the Davis-Yin method actually corresponds to a *two-splitting* of the gradient flow.

We now introduce two new accelerated variants of Algorithm 1. Let us consider analogous discretization approach but now applied to the accelerated gradient flow (19). Writting this system in first order form we have

$$\begin{aligned} \begin{bmatrix} \dot{x} \\ \dot{v} \end{bmatrix} &= \begin{bmatrix} v \\ -\eta(t)v - \nabla f(x) - \nabla g(x) - \nabla w(x) \end{bmatrix} \\ &= \underbrace{\begin{bmatrix} v/2 \\ -\eta(t)v - \nabla f(x) \end{bmatrix}}_{\varphi^{(1)}} + \underbrace{\begin{bmatrix} v/2 \\ -\nabla g(x) - \nabla w(x) \end{bmatrix}}_{\varphi^{(2)}} \end{aligned} \tag{42}$$

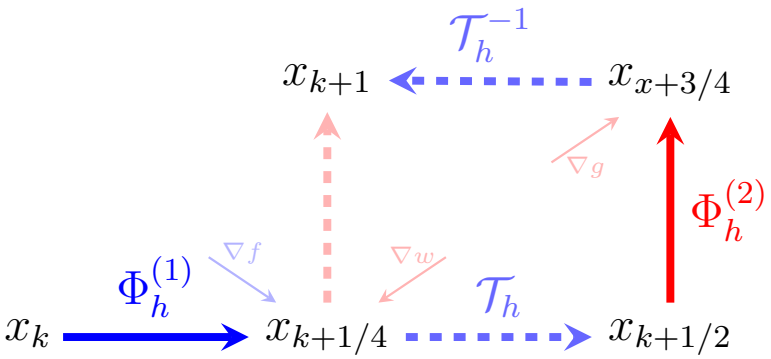

Figure 1: Illustration of the discretization underlying Algorithm 1. The starting point $x_k = x(t_k)$ is mapped to $x(t_{k+1}) = \Phi_h(x_k)$ under the continuous dynamics while to $x_{k+1} = \hat{\Phi}_h(x_k)$ under the numerical flow which is formed by the composition (49). We indicate over which points the gradients are computed which determines the implicit character of the discretization. By Proposition 3 we have that $x_{k+1} - x(t_{k+1}) = O(h^2)$.

whose splitting is equivalent to[5]

$$\ddot{x} + \eta(t)\dot{x} = -\nabla f(x), \qquad \ddot{x} = -\nabla g(x) - \nabla w(x). \tag{43}$$

Using (24), an implicit discretization of the first equation is $(x_{k+1/4} - \hat{x}_k)/h^2 = -\nabla f(x_{k+1/4})$. Letting $\lambda = h^2$ and using the resolvent (4) we thus have $x_{k+1/4} = J_{\lambda\nabla f}(\hat{x}_k)$. Employing (21) into the second equation under the semi-implicit discretization illustrated in Fig. 1 yields

$$(x_{k+3/4} - 2x_{k+1/4} + \hat{x}_k)/h^2 = -\nabla g(x_{k+3/4}) - \nabla w(x_{k+1/4}). \tag{44}$$

Note that, as in (37), we can define $x_{k+1/2} = \mathcal{T}_h(x_{k+1/4}) = 2x_{k+1/4} - \hat{x}_k$ so that this last update becomes $x_{k+3/4} = J_{\lambda\nabla g}\left(x_{k+1/2} - \lambda\nabla w(x_{k+1/4})\right)$. Finally, applying the inverse (41) we obtain $x_{k+1} = \mathcal{T}_h^{-1}(x_{k+3/4}) = x_{k+3/4} - (x_{k+1/4} - \hat{x}_k)$. Collecting these steps together we thus obtain the Algorithm 2. The factor γ_k defined in (23) accounts for both Nesterov and heavy ball type of accelerations. We thus introduced two accelerated variants of Davis-Yin method. To the best of our knowledge, these were not previously considered.

As previously mentioned, it is well-known that a two-splitting (12) yields a first order approximation to the continuous trajectory. However, let us show this explicitly since it will stress that the previous algorithms are indeed legit discretizations of the associated ODE.

[5]In writing (42) we defined the velocity $v \equiv \dot{x}$. As an artifice of splitting the two vector fields one has $v = 2\dot{x}$, which leads to $2(\ddot{x} + \eta\dot{x}) = -\nabla f(x)$ and $2\ddot{x} = -\nabla g(x) - \nabla w(x)$ instead of (43). This means that the original velocity was rescaled by twice its size, inconsistently with the original dynamics. Thus, one must ignore the factor of 2 as we did in (43), or absorb this factor in the discretization stepsize.

Algorithm 2 Accelerated variants of Davis-Yin. The variable $\hat{x}$ defined in (22) accounts for both Nesterov and heavy ball acceleration. When $f = 0$ this reduces accelerated forward-backward, while when $w = 0$ it reduces to accelerated variants of Douglas-Rachford.

for $k = 1, 2, \dots$ **do**
  $x_{k+1/4} \leftarrow J_{\lambda\nabla f}(\hat{x}_k)$
  $x_{k+1/2} \leftarrow 2x_{k+1/4} - \hat{x}_k$
  $x_{k+3/4} \leftarrow J_{\lambda\nabla g}\big(x_{k+1/2} - \lambda\nabla w(x_{k+1/4})\big)$
  $x_{k+1} \leftarrow \hat{x}_k + x_{k+3/4} - x_{k+1/4}$
  $\hat{x}_{k+1} \leftarrow x_{k+1} + \gamma_{k+1}(x_{k+1} - x_k)$
end for

Proposition 3. *Algorithm 1 is a first order integrator to the gradient flow* (18)*. Similarly, Algorithm 2 is a first order integrator to the accelerated gradient flow* (19)*.*

Proof. We consider Algorithm 2 in detail since it is more involved than Algorithm 1. It is well-known that for a differentiable function f one has [31, 51]

$$J_{\lambda\nabla f}(x) = x - \lambda\nabla f(x) + \mathcal{O}(\lambda^2). \tag{45}$$

Through a Taylor expansion, the first three updates of Algorithm 2 become

$$x_{k+1/4} = \hat{x}_k - \lambda\nabla f(\hat{x}_k) + \mathcal{O}(\lambda^2), \tag{46a}$$
$$x_{k+1/2} = \hat{x}_k - 2\lambda\nabla f(\hat{x}_k) + \mathcal{O}(\lambda^2), \tag{46b}$$
$$x_{k+3/4} = \hat{x}_k - 2\lambda\nabla f(\hat{x}_k) - \lambda\nabla g(\hat{x}_k) - \lambda\nabla w(\hat{x}_k) + \mathcal{O}(\lambda^2), \tag{46c}$$

implying that the fourth update yields

$$\frac{x_{k+1} - \hat{x}_k}{\lambda} = -\nabla f(\hat{x}_k) - \nabla g(\hat{x}_k) - \nabla w(\hat{x}_k) + \mathcal{O}(\lambda^2). \tag{47}$$

Setting $\lambda = h^2$, using (24), and noticing that $\hat{x}_k = x(t_k) + \mathcal{O}(h)$ we obtain

$$\ddot{x}(t_k) + \eta(t_k)\dot{x}(t_k) = -\nabla f(x(t_k)) - \nabla g(x(t_k)) - \nabla w(x(t_k)) + \mathcal{O}(h), \tag{48}$$

where $x_k \equiv x(t_k)$. This means that Algorithm 2 reproduces the ODE (19) up to an error $\mathcal{O}(h)$. But this automatically implies that the numerical estimate is close to the continuous trajectory[6] up to $\mathcal{O}(h^2)$, therefore the method is first order accurate. Similar arguments apply to Algorithm 1 in relation to the gradient flow (18) by using relation (20). □

[6] Suppose we write the system in first order form $\dot{z} = F(z)$, for some $z = z(t)$. If the numerical map reproduces $\dot{z}_k = F(z_k) + \mathcal{O}(h)$ (we denote $z_k = z(t_k)$) then $z_{k+1} = z_k + hF(z_k) + \mathcal{O}(h^2)$. Now from the continuous dynamics we have $z(t_k + h) = z(t_k) + hF(z(t_k)) + \mathcal{O}(h^2)$, thus $\|z_{k+1} - z(t_k + h)\| = \mathcal{O}(h^2)$.

Since the above splitting scheme is not balanced, it is not a priori obvious if the abvoe discretizations preserve steady states of the associated ODE. To see that this is indeed the case, note that both Algorithm 1 and Algorithm 2 can be seen as a result of the flow composition

$$\hat{\Phi}_h = \mathcal{T}_h^{-1} \circ \Phi_h^{(2)} \circ \mathcal{T}_h \circ \Phi_h^{(1)}. \tag{49}$$

Both methods thus perform the respective iterations $x_{k+1} = \hat{\Phi}_h(x_k)$ and $x_{k+1} = \hat{\Phi}_h(\hat{x}_k)$. We can thus analyse both algorithms simultanously to conclude the following.

Proposition 4. *Both Algorithm 1 and Algorithm 2 preserve steady states of the respective ODEs* (18) *and* (19)*.*

Proof. From the fourth update of Algorithms 1 and 2 it is clear that

$$\begin{aligned}
\hat{\Phi}_h &= I + J_{\lambda\nabla g} \circ (2J_{\lambda\nabla f} - I - \lambda\nabla w \circ J_{\lambda\nabla f}) - J_{\lambda\nabla f} \\
&= I + J_{\lambda\nabla g} \circ (C_{\lambda\nabla f} - \lambda\nabla w \circ J_{\lambda\nabla f}) - \tfrac{1}{2}(C_{\lambda\nabla f} + I) \\
&= \tfrac{1}{2}I - \tfrac{1}{2}C_{\lambda\nabla f} + \tfrac{1}{2}\left(C_{\lambda\nabla g} + I\right) \circ (C_{\lambda\nabla f} - \lambda\nabla w \circ J_{\lambda\nabla f}) \\
&= \tfrac{1}{2}I + \tfrac{1}{2}C_{\lambda\nabla g} \circ (C_{\lambda\nabla f} - \lambda\nabla w \circ J_{\lambda\nabla f}) - \tfrac{1}{2}\lambda\nabla w \circ J_{\lambda\nabla f}
\end{aligned} \tag{50}$$

which is precisely the operator $\mathcal{P}$ in (27), thus the claim follows from Lemma 2. □

Note that the above result does not imply any convergence guarantees for the discrete algorithms. It only implies that, *if the algorithm converges*, then it must converge to a fixed point of the operator $\hat{\Phi}_h$, and as a consequence such a point must be a steady state of the associated continuous dynamical system.

3.1.1 Douglas-Rachford

When $w = 0$, Algorithm 1 reduces to the well-known Douglas-Rachford algorithm [17, 21]. The same is true for Algorithm 2 which reduces to accelerated variants of Douglas-Rachford. In particular, when the damping (23a) related to Nesterov acceleration is used, the resulting method was proposed in [52]. We are not aware of previous considerations of the heavy ball variant of Douglas-Rachford that arises as a particular case of Algorithm 2. Our previous results automatically imply that Douglas-Rachford is a discretization of the gradient flow, while its accelerated variants are discretization of the accelerated gradient flow. Moreover, these are first order integrators. Note that they preserve steady states of the underlying ODE since (50) becomes $\frac{1}{2}I + \frac{1}{2}C_{\lambda\nabla g} \circ C_{\lambda\nabla f}$ whose fixed points are zeros of $\nabla f + \nabla g$. Douglas-Rachford algorithm has been extensively studied in the literature. For some recent results on this interesting algorithm, see [53] and references therein.

3.1.2 Forward-Backward

When $f = 0$ we obtain from Algorithm 1 the forward-backward splitting method [17–19]. Similarly, Algorithm 2 provide its accelerated variants

$$x_{k+1} = J_{\lambda\nabla g}\big(\hat{x}_k - \lambda\nabla w(\hat{x}_k)\big), \tag{51a}$$

$$\hat{x}_{k+1} = x_{k+1} + \gamma_{k+1}(x_{k+1} - x_k). \tag{51b}$$

From an ODE perspective, this is actually not a splitting method but only a semi-implicit discretization (note that the first equation in (35) is absent). Nevertheless, these are still first order integrators. Note that the operator (50) reduces to $J_{\lambda\nabla g} \circ (I - \lambda\nabla w)$ whose fixed points are zeros of $\nabla g + \nabla w$, and vice-versa. The algorithm (51) was recently studied in detail [11] and when Nesterov acceleration is used (23a) it was shown to have a convergence rate of $\mathcal{O}(1/k^2)$ for convex functions, matching the continuous rate of Table 1. In signal processing literature the forward-backward splitting (non accelerated) is known as ISTA [45] and its Nesterov accelerated variant is known as FISTA [54]. Both are widely used solvers.

3.2 Extensions of ADMM via Rebalanced Splitting

We now show that ADMM corresponds to a different kind of splitting, namely a rebalanced splitting (see Section 2). We will actually construct new algorithms that are extensions of ADMM, together with accelerated variants, that can be applied to problems in the form (34) (the original ADMM can only be applied with $w = 0$).

Consider the gradient flow (18) with vector fields $\varphi^{(1)} = -\nabla f - \nabla w$ and $\varphi^{(2)} = -\nabla g$. According to the balanced splitting approach we thus have

$$\dot{x} = -\nabla f(x) - \nabla w(x) + c, \qquad \dot{x} = -\nabla g(x) - c, \tag{52}$$

where c is the balance coefficient (see (16) and (17)). An implicit discretization is equivalent to approximating an integral in (16) by its upper limit, while an explicit discretization is equivalent to using the lower limit. We choose a forward-backward discretization for the first equation in (52) and an implicit discretization for the second one. Thus,

$$\int_{t_k}^{t_k+h} \varphi^{(1)} dt \approx -h\nabla f(x_{k+1/2}) - h\nabla w(x_k), \tag{53a}$$

$$\int_{t_k}^{t_k+h} \varphi^{(2)} dt \approx -h\nabla g(x_{k+1}). \tag{53b}$$

Using the resolvent (4) to solve equations (16), with stepsize $h = \lambda$, and further replacing the resulting updates into (17) we then obtain Algorithm 3. Note that in practice w must

Algorithm 3 Extension of ADMM from a rebalanced splitting described in (16) and (17). Besides introducing the function w so that this methods solves problems of the form (34), another difference compared to standard ADMM is the dual update.

for $k = 1, 2, \dots$ **do**
  $x_{k+1/2} \leftarrow J_{\lambda\nabla f}\left(x_k - \lambda\nabla w(x_k) + \lambda c_k\right)$
  $x_{k+1} \leftarrow J_{\lambda\nabla g}(x_{k+1/2} - \lambda c_k)$
  $c_{k+1} \leftarrow c_k + \frac{1}{\lambda}\left(\frac{1}{2}x_{k+1} + \frac{1}{2}x_k - x_{k+1/2}\right)$
end for

Algorithm 4 Accelerated variants of Algorithm 3 from a rebalanced splitting approach. This accounts to both Nesterov and heavy ball acceleration according to (23).

for $k = 1, 2, \dots$ **do**
  $x_{k+1/2} \leftarrow J_{\lambda\nabla f}\left(\hat{x}_k - \lambda\nabla w(\hat{x}_k) + \lambda c_k\right)$
  $x_{k+1} \leftarrow J_{\lambda\nabla g}(x_{k+1/2} - \lambda c_k)$
  $c_{k+1} \leftarrow c_k + \frac{1}{\lambda}\left(\frac{1}{2}x_{k+1} + \frac{1}{2}x_k - x_{k+1/2}\right)$
  $\hat{x}_{k+1} \leftarrow x_{k+1} + \gamma_{k+1}(x_{k+1} - x_k)$
end for

be differentiable, however f and g are allowed to be nonsmooth. By construction, this discretization is guaranteed to preserve steady states of the continuous system through both equations in (52). Algorithm 3 is new to the best of our knowledge, however it has similarities with ADMM [23, 24, 26]. More precisely, when $w = 0$ the first two updates are the same as the ones of ADMM but the third update slightly different (c is usually called the dual variable).

Let us now introduce an accelerated version of Algorithm 3. The same rebalanced splitting applied to the accelerated gradient flow (42) yields

$$\ddot{x} + \eta(t)\dot{x} = -\nabla f(x) - \nabla w(x) + c, \qquad \ddot{x} = -\nabla g(x) - c. \tag{54}$$

With $\lambda = h^2$, a forward-backward discretization of the first equation results in $x_{k+1/2} - \hat{x}_k = -\lambda\nabla f(x_{k+1/2}) - \lambda\nabla w(\hat{x}_k) + \lambda c_k$. For the second equation, denote the endpoint by $\tilde{x}_{k+1}$. An implicit discretization is $\tilde{x}_{k+1} - 2x_{k+1/2} + \hat{x}_k = -\lambda\nabla g(x_{k+1}) - \lambda c_k$. If we denote $x_{k+1} \equiv \tilde{x}_{k+1} - (x_{k+1/2} - \hat{x}_k)$ this is equivalent to $x_{k+1} - x_{k+1/2} = -\lambda\nabla g(x_{k+1}) - \lambda c_k$. The balance equation remains unchanged. Therefore, employing the resolvent (4) we obtain Algorithm 4, which readily incorporates Nesterov and heavy ball type of acceleration through (23).

An important characteristic of the above rebalanced splitting it that both individual equations in (52) separately preserve steady states since $c_\infty = \frac{1}{2}\big(\varphi^{(2)}(x_\infty) - \varphi^{(1)}(x_\infty)\big)$ (recall from (15)). The updates of the balance coefficient c play an important role in the stability of the method [29]. If $\varphi^{(2)}$ is much more "stiff" than $\varphi^{(1)}$ this rebalanced splitting may be

Algorithm 5 Extension of ADMM for solving problems in the form (34). This method is a rebalanced splitting. The standard ADMM is recovered when $w = 0$.

1: **for** $k = 1, 2, \dots$ **do**
2: $\quad x_{k+1/2} \leftarrow J_{\lambda\nabla f}(x_k - \lambda\nabla w(x_k) + \lambda c_k)$
3: $\quad x_{k+1} \leftarrow J_{\lambda\nabla g}(x_{k+1/2} - \lambda c_k)$
4: $\quad c_{k+1} \leftarrow c_k + \frac{1}{\lambda}\left(x_{k+1} - x_{k+1/2}\right)$
5: **end for**

unstable for large stepsizes [29]. In an optimization context where g is usually a regularizer, it may be desirable to preserve steady states only through the first flow of (52) or (54). This can be accomplished by choosing the balance coefficient as

$$c_\infty = \varphi^{(2)}(x_\infty) \tag{55}$$

implying that $\dot{x}_\infty = \left(\varphi^{(1)} + \varphi^{(2)}\right)(x_\infty) = 0$ for the first equation in (13), so that steady states are preserved. The second equation becomes simply the identity $\dot{x}_\infty = \left(\varphi^{(2)} - \varphi^{(2)}\right)(x_\infty) = 0$. Note that with this approach the only difference in the discretization schemes leading to Algorithm 3 and Algorithm 4 lies on the balance equation. Now, alternatively to (17) we consider the average

$$c_{k+1} = \frac{1}{h}\int_{t_k}^{t_k+h} \varphi^{(2)} dt = \frac{1}{h}\left(x_{k+1} - x_{k+1/2} + hc_k\right) \tag{56}$$

where we used (16b) to replace the integral by the previous iterates. Incorporating the modification in the balance equation we thus obtain the Algorithm 5. Note that this algorithm recovers ADMM [23, 24, 26] exactly[7] in the particular case where $w = 0$. Algorithm 5 is a generalization of ADMM for solving problems in the form (34).

We stress the interesting connection with ODEs where Algorithm 5 together with the well-known ADMM algorithm, proposed since the 70's, actually correspond to a rebalanced splitting discretization [29] of the gradient flow. Note also that they preserve steady states of the continuous dynamical system but only through the flow of the first equation in (52).

An accelerated variant of Algorithm 6 can be readily obtained by changing the balance equation of Algorithm (4). For further reference, we explicitly write the resulting udpates in Algorithm 6. Note that when $w = 0$ we have an accelerated version of standard ADMM, however this version is different than the Fast ADMM proposed in [55] where besides accelerating the x variable it also accelerates the balance coefficient c, i.e. there is an extra update in the same form as line 5 of Algorithm 6 with $x \to c$. Note also that [55] only accounts

[7] One can identify $c_k \equiv -u_k$ as the dual variable which is a Lagrange multiplier in the original derivation of the algorithm from an augmented Lagrangian approach [26].

Algorithm 6 Accelerated variants of Algorithm 5. This accounts for Nesterov and heavy ball acceleration through (23). When $w = 0$ this yields an accelerated variant of ADMM.

1: **for** $k = 1, 2, \dots$ **do**
2: $\quad x_{k+1/2} \leftarrow J_{\lambda\nabla f}(\hat{x}_k - \lambda\nabla w(\hat{x}_k) + \lambda c_k)$
3: $\quad x_{k+1} \leftarrow J_{\lambda\nabla g}(x_{k+1/2} - \lambda c_k)$
4: $\quad c_{k+1} \leftarrow c_k + \frac{1}{\lambda}\left(x_{k+1} - x_{k+1/2}\right)$
5: $\quad \hat{x}_{k+1} \leftarrow x_{k+1} + \gamma_{k+1}(x_{k+1} - x_k)$
6: **end for**

for Nesterov acceleration with damping (23a). The continuous limit of Fast ADMM [55] was considered in more generality by us in [15, 16] and also corresponds to a second order ODE with similar form as (19) (with $w = 0$), however in this case the method is not a rebalanced splitting discretization such as in our derivation leading to Algorithm 6.

Next, we consider how accurate the above extensions of ADMM and accelerated variants reproduce the continuous dynamics.

Proposition 5. *Algorithm 3 and Algorithm 5 are first order integrators to the composite gradient flow* (18)*. Similarly, Algorithm 4 and Algorithm 6 are first order integrators to the composite accelerated gradient flow* (19)*.*

Proof. We consider the accelerated variants since analogous arguments apply to the nonaccelerated ones. As discussed in the proof of Proposition 3, it is enough to verify that the discretization reproduces the ODE up to an error $O(h)$. Note that first two updates of Algorithm 4 and Algorithm 6 are the same. Using the approximation (45) and Taylor expansions we have

$$x_{k+1/2} = \hat{x}_k - \lambda\nabla w(\hat{x}_k) + \lambda c_k - \lambda\nabla f(\hat{x}_k) + \mathcal{O}(\lambda^2), \tag{57a}$$

$$x_{k+1} = \hat{x}_k - \lambda\nabla w(\hat{x}_k) - \lambda\nabla f(\hat{x}_k) - \lambda\nabla g(\hat{x}_k) + \mathcal{O}(\lambda^2). \tag{57b}$$

This last update implies

$$\frac{x_{k+1} - \hat{x}_k}{\lambda} = -\nabla f(\hat{x}_k) - \nabla g(\hat{x}_k) - \nabla w(\hat{x}_k) + \mathcal{O}(\lambda). \tag{58}$$

From (22) one can concludes that $\hat{x}_k = x_k + O(h)$, where $x_k = x(t_k)$, and using (24) with $\lambda = h^2$ the above implies that we reproduce (19) up to $O(h)$. □

3.3 Tseng Splitting

The last proximal splitting method that remains to be considered is the forward-backward-forward splitting proposed by Tseng [20]. We do this now for completeness. Let us set

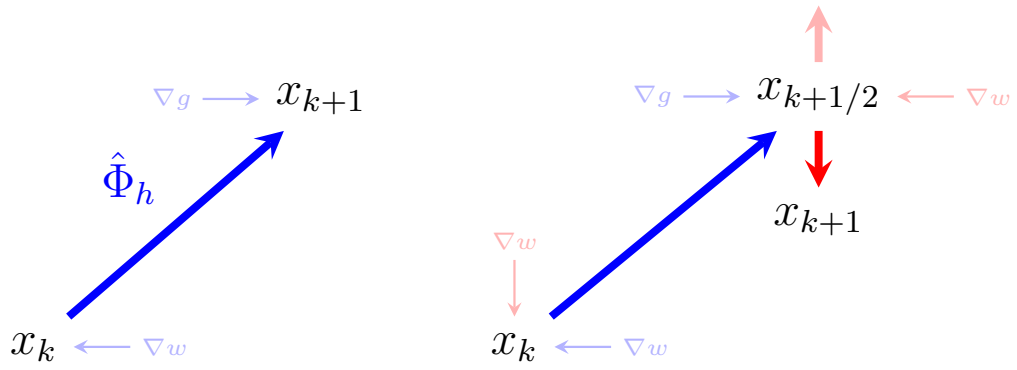

Figure 2: Illustration of forward-backward (left) vs. Tseng splitting (right). *Left:* This is just a semi-implicit Euler discretization to the gradient flow leading to the forward-backward method and its accelerated variant (51). *Right:* Tseng splitting consists of adding a perturbation on top of forward-backward. The perturbation is small being of $\mathcal{O}(\lambda^3)$.

$f = 0$ in the composite gradient flow (18) so that we are solving problems of the form $\min_x g(x) + w(x)$, where w is differentiable. This method consists of a simple modification of the forward-backward splitting so that it converges under weaker assumptions [20]. To see the connections with ODEs, let us consider splitting the gradient flow as

$$\dot{x} = -\nabla g(x) - \nabla w(x), \qquad \dot{x} = \nabla w(x) - \nabla w(x). \tag{59}$$

Note that we added $\nabla w(x) - \nabla w(x) = 0$ to the right hand side before splitting. The purpose of the second equation in (59) is to enforce $\dot{x} \approx 0$ and we believe this is the reason for the improved stability of Tseng's method compared to forward-backward. Proceeding, the first equation in (59) is discretized through forward-backward Euler yielding $(x_{k+1/2} - x_k)/h = -\nabla g(x_{k+1/2}) - \nabla w(x_k)$, where $h = \lambda$ is the stepsize. For the second equation in (59), consider $(x_{k+1} - x_{k+1/2})/h = \nabla w(x_k) - \nabla w(x_{k+1/2})$. Note that x_{k+1} can can be seen as a perturbation of $x_{k+1/2}$ given by the difference of the gradients at the two nearby points x_k and $x_{k+1/2}$. This perturbation vanishes when both points get very close to each other. The overall discretization scheme is illustrated in Fig. 2. Finally, using the resolvent (4) we obtain Algorithm 7. Note that it requires two gradient computations on w per iteration, however when the gradient has a simple closed form this extra cost can be negligible.

To gain some insight into the perturbation, assume that all functions are smooth, use (45) and a Taylor expansion to obtain

$$\begin{aligned} \lambda\nabla w(x_k) - \lambda\nabla w(x_{k+1/2}) &= \lambda\nabla w - \lambda\nabla w\big(x_k - \lambda\nabla w - \lambda\nabla g + \mathcal{O}(\lambda^2)\big) \\ &= -\lambda^3(\nabla w + \nabla g)^T(\nabla^2 w)(\nabla w + \nabla g) + \mathcal{O}(\lambda^5) \end{aligned} \tag{60}$$

Algorithm 7 Tseng splitting method [20].

for $k = 1, 2, \dots$ **do**
  $x_{k+1/2} \leftarrow J_{\lambda \nabla g}(x_k - \lambda \nabla w(x_k))$
  $x_{k+1} \leftarrow x_{k+1/2} - \lambda(\nabla w(x_{k+1/2}) - \nabla w(x_k))$
end for

Algorithm 8 Accelerated version of Tseng splitting given in Algorithm 7.

for $k = 1, 2, \dots$ **do**
  $x_{k+1/2} \leftarrow J_{\lambda \nabla g}\left(\hat{x}_k - \lambda \nabla w(\hat{x}_k)\right)$
  $x_{k+1} \leftarrow x_{k+1/2} - \lambda\left(\nabla w(x_{k+1/2}) - \nabla w(\hat{x}_k)\right)$
  $\hat{x}_{k+1} \leftarrow x_{k+1} + \gamma_{k+1}\left(x_{k+1} - x_k\right)$
end for

where in this expression all gradients and the Hessian $\nabla^2 w$ are being computed at the point x_k, unless explicitly specified. We thus see that the perturbation is of the order λ^3 and moreover it captures the curvature of the function w.

Let us now propose an accelerated variant of Algorithm 7 which arises as a natural discretization of the accelerated gradient flow (19) with $f = 0$. Consider splitting the system

$$\begin{bmatrix} \dot{x} \\ \dot{v} \end{bmatrix} = \underbrace{\begin{bmatrix} v/2 \\ -\eta(t)v - \nabla g(x) - \nabla w(x) \end{bmatrix}}_{\varphi^{(1)}} + \underbrace{\begin{bmatrix} v/2 \\ \nabla w(x) - \nabla w(x) \end{bmatrix}}_{\varphi^{(2)}}, \tag{61}$$

which leads to

$$\ddot{x} + \eta(t)\dot{x} = -\nabla g(x) - \nabla w(x), \qquad \ddot{x} = \nabla w(x) - \nabla w(x). \tag{62}$$

Setting $\lambda = h^2$, a forward-backward Euler discretization of the first equation yields the update $x_{k+1/2} = J_{\lambda \nabla g}\left(\hat{x}_k - \lambda \nabla w(\hat{x}_k)\right)$. As before, for the second equation we use (21) to obtain $(\tilde{x}_{k+1} - 2x_{k+1/2} + \hat{x}_k)/\lambda = \nabla w(\hat{x}_k) - \nabla w(x_{k+1/2})$. Defining the new variable $x_{k+1} \equiv \tilde{x}_{k+1} - (x_{k+1/2} - \hat{x}_k)$ and collecting these steps, we are left with Algorithm 8. This constitute accelerated versions of Algorithm 7 that that can be of Nesterov and heavy ball types (see (23)). We believe these accelerated variants of Tseng splitting were not previously considered in the literature.

Using exactly the same approach as in the proof of Proposition 3 and Proposition 5 we conclude that both (8) and (7) are first order integrators. We state this result below and ommit the proof since the argument was already provided.

Proposition 6. *Tseng splitting given by Algorithm 7 is a first order integrator to the gradient flow* (18) *(with $f = 0$). Also, the accelerated variant of this method given in Algorithm 8 is a first order integrator to the accelerated gradient flow* (19).

Regarding preservation of steady states it is easy to show the following.

Proposition 7. *Both Algorithm 7 and Algorithm 8 preserve steady states of their underlying gradient flow.*

Proof. From the second updates of Algorithm 7 and Algorithm 8 we see that these are equivalent to finding fixed points of

$$\hat{\Phi}_h = (I - \lambda\nabla w) \circ J_{\lambda\nabla g} \circ (I - \lambda\nabla w) + \lambda\nabla w. \tag{63}$$

It is immediate that$(\nabla g + \nabla w)(\bar{x}) = 0 \iff \hat{\Phi}_h(\bar{x}) = \bar{x}$, which finishes the proof. □

4 Monotone Operators

We previously assumed that all functions were differentiable implying that the associated operators were single-valued. This allowed us to provide direct derivations without worrying about technical details. Since proximal algorithms can naturally be generalized to the level of monotone operators, we now discuss how our previous analysis apply in this context.

First, we briefly introduce the necessary concepts about monotone operators. (The reader is referred to [31] for details.) Let H be a Hilbert space with inner product $\langle\cdot,\cdot\rangle : H \times H \to H$. A monotone operator A is a multi-valued map $A : H \rightrightarrows H$, with $\operatorname{dom} A \equiv \{x \in H \mid Ax \neq 0\}$, if and only if

$$\langle Ay - Ax, y - x\rangle \geq 0 \qquad \text{for all } x, y \in \operatorname{dom} A. \tag{64}$$

A monotone operator is said to be *maximal* if no enlargement of its graph is possible. Every monotone operator admits a maximal extension. The resolvent of A is defined by (4) and it is possible to show that it is a single-valued map, i.e. $J_{\lambda A} : H \to H$. Moreover, $x^\star \in \operatorname{zer} A \iff J_{\lambda A}x^\star = x^\star$, i.e. zeros of A are fixed points of $J_{\lambda A}$ and vice-versa. An important ingredient is the *Yosida regularization* of A defined as

$$A_\mu \equiv \mu^{-1}(I - J_{\mu A}) \qquad (\mu > 0). \tag{65}$$

This operator is single-valued since $J_{\lambda A}$ is single-valued (for a maximal monotone operator A) and Lipschitz continuous. Moreover, $0 \in A\bar{x} \iff 0 = A_\mu\bar{x}$, 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 [31]. Often one is faced with the resolvent of the Yosida regularization, i.e. $J_{\lambda A_\mu} = (I - \lambda A_\mu)^{-1}$. The following useful identity allows one to express this in terms of the original operator A [31]:

$$J_{\lambda(A_\mu)} = I + \frac{\lambda}{\mu + \lambda}(J_{(\mu+\lambda)A} - I). \tag{66}$$

Importantly, in the limit $\mu \downarrow 0$ we see that (66) recovers the original resolvent $J_{\lambda A}$.

Let us now consider an approach through differential inclusions (see [56] for background). Consider the first order differential inclusion

$$\dot{x} \in -Ax - Bx - Cx \tag{67}$$

and also the second order differential inclusion

$$\ddot{x} + \eta(t)\dot{x} \in -Ax - Bx - Cx, \tag{68}$$

where A, B, C are maximal monotone and C is further assumed to be single-valued. The previous discretizations of the gradient flow (18), and the accelerated gradient flow (19), extend naturally to (67) and (68) thanks to the fact that the resolvent is single-valued. We believe it is enough to illustrate this through an example. Thus, for simplicity consider (67) with $A = 0$. Under a forward-backward Euler discretization (see Fig. 2) with stepsize $h = \lambda$ one has $(x_{k+1} - x_k)/\lambda \in -Bx_{k+1} - Cx_k$, or equivalently $(I + \lambda B)x_{k+1} \ni x_k - Cx_k$. Using the resolvent (4) and the fact that for a maximal monotone operator it is a single-valued map, this implies that

$$x_{k+1} = J_{\lambda B}(x_k - \lambda C x_k). \tag{69}$$

The same reasoning applies to (68) by using (24). Note that this is the operator analog of the forward-backward method (51).

The reader can easily convince himself that all the previous approaches related to Davis-Yin, extensions of ADMM and Tseng splitting, and their accelerated variants, hold in a similar manner for maximal monotone operators through discretizing the differential inclusions (67) and (68).

Let us now consider a second approach through regularized monotone operators. By employing the Yosida regularization (65) we can consider the ODEs

$$\dot{x} = -A_\mu x - B_\mu x - Cx \tag{70}$$

and

$$\ddot{x} + \eta(t)\dot{x} = -A_\mu x - B_\mu x - Cx \tag{71}$$

in place of (67) and (68), respectively. Since the Yosida regularization is Lipschitz continuous, these ODEs admit a unique global solution. Had instead we considered a single operator, or equivalently $(A + B + C)_\mu$, the associated ODEs would have the same steady states as the original differential inclusions (note that $\operatorname{zer} A = \operatorname{zer} A_\mu$). However, one would not be able to employ splitting methods. On the other hand, for an operator sum we have that

$$\operatorname{zer}(A + B) \neq \operatorname{zer}(A_\mu + B_\mu) \tag{72}$$

so that the ODEs (70) and (71) do not have steady states compatible with zeros of $A+B+C$. Nevertheless, after discretizing these ODEs, one just needs to take the limit $\mu \downarrow 0$ to recover iterates aimed at finding zeros of $A+B+C$. This procedure will give exactly the same iterates as if one discretizes the original differential inclusions (67) or (68). In this procedure, the identity (66) plays an important role.

Let us provide a concrete example to illustrate the point of the previous paragraph. Consider (70) with $A = 0$ for simplicity. Performing a forward-backward Euler discretization, i.e. $(x_{k+1}-x_k)/\lambda = -B_\mu x_{k+1} - Cx_k$, we obtain $(I+\lambda B_\mu)x_{k+1} = x_k - Cx_k$. Using the resolvent (4) this is equivalent to $x_{k+1} = J_{\lambda B_\mu}(x_k - \lambda C x_k)$. We can now use the identity (66) to obtain

$$x_{k+1} = \frac{1}{\mu+\lambda}\left(\mu I + \lambda J_{(\mu+\lambda)B}\right) \circ (I - \lambda C)x_k. \tag{73}$$

Finally, in the limit $\mu \downarrow 0$ this yields precisely (69).

To make the above concepts more familiar in an optimization context, consider the case where $A = \partial f$ is the subdifferential of a nonsmooth function f. The Yosida regularization (66) becomes

$$\nabla f_\mu(x) = \mu^{-1}(x - \mathrm{prox}_{\mu f}(x)) \tag{74}$$

which is the gradient of the Moreau envelope $f_\mu(x) \equiv \min_y \left(f(y) + \frac{1}{2\mu}\|y-x\|^2\right)$. The Moreau envelope is always differentiable and has the same minimizers as f. Recently, convergence analysis of the accelerated gradient flow with regularized monotone operators and Moreau envelopes were considered [10].

5 Numerical Experiments

The goal of this section is to show that the accelerated variants of each respective algorithm improves over the standard (nonaccelerated) formulation. Since most of the new algorithms were obtained for the general problem (34) that involves three functions, we focus on Davis-Yin and extensions of ADMM. To this end, we consider a matrix completion problem where the entries of the matrix are restricted to lie on a certain range. Given a matrix $M \in \mathbb{R}^{n\times m}$ assumed to have *low rank*, suppose we only have access to some of its entries indexed by a set Ω, i.e. our input data is $M_{\text{obs}} = \mathcal{P}_\Omega(M)$ where $\mathcal{P}_\Omega$ projects onto the support of observed entries: $\mathcal{P}_\Omega(M)_{ij} = M_{ij}$ if $(i,j) \in \Omega$ and $\mathcal{P}_\Omega(M)_{ij} = 0$ otherwise. The goal is to reconstruct M using only knowledge of M_{obs}, i.e. we wish to complete the missing entries $(i,j) \notin \Omega$ of the matrix M_{obs}. A possible approach is to consider the rank minimization problem

$$\min_{X\in\mathbb{R}^{n\times m}} \mathrm{rank}(X) \quad \text{such that } \mathcal{P}_\Omega(X) = \mathcal{P}_\Omega(M). \tag{75}$$

Unfortunately, there is no hope to solve this problem since it is nonconvex and NP-hard. Alternatively, the convex relaxation

$$\min_{X\in\mathbb{R}^{m\times n}} \|X\|_* \quad \text{such that } \mathcal{P}_\Omega(X) = \mathcal{P}_\Omega(M) \tag{76}$$

is convex and feasible [57]. Above, $\|X\|_*$ is the nuclear norm which is the sum of singular values of X. We consider still an alternative form of (76), and further impose a box constraint $\ell_1 \leq X_{ij} \leq \ell_2$. Thus, we consider the optimization problem

$$\min_{X\in\mathbb{R}^{n\times m}} \underbrace{\mu\|X\|_*}_{f(X)} + \underbrace{\mathbb{I}_{[\ell_1,\ell_2]}(X)}_{g(X)} + \underbrace{\tfrac{1}{2}\|\mathcal{P}_\Omega(X) - \mathcal{P}_\Omega(M)\|_F^2}_{w(X)} \tag{77}$$

where $\|\cdot\|_F$ is the Frobenius norm and $\mathbb{I}_{[\ell_1,\ell_2]}(X) = 0$ if $\ell_1 \leq X_{ij} \leq \ell_2$ for all i, j, and $\mathbb{I}_{[\ell_1,\ell_2]}(X) = \infty$ otherwise is the indicator function. The parameter μ controls the strength of the nuclear norm and large μ imposes low rank solutions [57]. We will measure the performance of our algorithms through the relative error

$$E = \frac{\|X^\star - M\|_F}{\|M\|_F} \tag{78}$$

where $X^\star$ is a numerical solution to (77). We naturally assume that we know the ground truth M. The stopping criteria for our algorithms will be

$$\frac{\|X_{k+1} - X_k\|_F}{\max\big(1, \|X_k\|_F\big)} \leq \epsilon \tag{79}$$

for some predefined tolerance ϵ, or when maximum number of iterations $k \leq k_{\max}$ is reached. Here X_k is the solution estimate at iteration k. This is a very reasonable condition for convergence and was previously employed with success for this type of problems [58].

Note that problem (77) can be readily implemented in the previous algorithms using the closed form solution to the proximal operator of f given by [57]

$$\text{prox}_{\lambda\mu\|\cdot\|_*}(X) = U D_{\lambda\mu}(\Sigma) V^T \tag{80}$$

where $X = U\Sigma V^T$ is the singular value decomposition (SVD) and $D_\tau(\Sigma)_{ii} = \max(\Sigma_{ii} - \tau, 0)$. The proximal operator of g is just the projection

$$\big(\text{prox}_{\lambda\mathbb{I}_{[\ell_1,\ell_2]}}(X)\big)_{ij} = \max(\ell_1, \min(X_{ij}, \ell_2)), \tag{81}$$

and the gradient of w simply $\nabla w(X) = \mathcal{P}_\Omega(X - M)$.

Annealing Besides running each algorithm with a fixed μ we will also implement an annealing schedule on this parameter, which can help improve the accuracy of the solution. We wish to verify if accelerated variants of the algorithms can also bring benefits in this scenario. The procedure is simple. Given a decreasing sequence $\mu_1 \geq \mu_2 \geq \cdots \geq \mu_L = \bar{\mu} > 0$ we run each algorithm exactly as before with each μ_j $(j = 1, \ldots, L)$ and use the solution obtained with μ_j as a warm start for the next run with parameter μ_{j+1}. The other parameters are kept fixed across different runs. In optimization, such an approach was proposed in [59] and used in matrix completion by [58]. We use exactly the same procedure as [58] which is as follows. We start with $\mu_0 = \eta \|M_{\text{obs}}\|_F$ then decrease the stepsize according to the schedule $\mu_{j+1} = \max(\eta\mu_j, \bar{\mu})$, for a given $\eta < 1$, until reaching $\bar{\mu}$.

Unless mentioned otherwise, in our experiments we choose the initial condition, stepsize, and other parameters as

$$\begin{gathered} X_0 = 0, \quad \lambda = 1, \quad \eta = 1/4, \quad \epsilon = 10^{-6}, \quad k_{\max} = 600, \quad \bar{\mu} = 10^{-8} \\ \ell_1 = \min(M_{\text{obs}}) - \sigma, \quad \ell_2 = \max(M_{\text{obs}}) + \sigma, \quad r_{\text{nest}} = 20, \quad r_{\text{hb}} = -\log 0.65, \end{gathered} \tag{82}$$

where σ is the standard deviation of the entries of M_{obs}. We observed that overdamping the accelerated systems gave faster convergence in some cases (especially with annealing), whence the choices of damping coefficients r_{nest} for Nesterov and r_{hb} for heavy ball (see (23)). The parameter μ will be specified in each case. To alleviate the computational burden of computing SVDs, we always compute a truncated SVD with 150 components (this is enough since in our examples M will have rank $r = 70$). Our experiments were conducted on a personal MacBook Pro with Intel Core i5 2.3GHz 4 cores processor and 16GB of RAM, and the implementations were done in Python 3.

Let us mention that a matrix of rank r has $d = r(n + m - r)$ degrees of freedom. Thus if we sample a matrix $M \in \mathbb{R}^{n\times m}$ uniformly at random with a sampling rate s we have $p = snm$ measurements. A quantification of the hardness of the problem is thus given by d/p [57]. When this ratio is $d/p > 1$ the problem does not have a single solution thus recovery is impossible.

To consider a concrete situation we choose a grayscale image represented by a $n \times m = 974 \times 1194$ real matrix with entries in the range $[0, 1]$. We enforce low rank by computing a truncated SVD to obtain the image shown in Fig. 3 (left) which has rank $r = 70$. By sampling its entries uniformly at random we obtain the input data shown in Fig. 3 (middle) where we choose a sampling rate of $s = 0.5$, which gives $d/p = 0.260$. We applied Algorithms 1 and 2, Algorithms 5 and 6, and Algorithms 3 and 4 with fixed and annealed μ. With a fixed $\mu = 0.05$ an example of the recovered image is shown in Fig. 3 (right). All the algorithms performed very similarly, however Davis-Yin showed a marginal improvement in the relative error. All methods recovered the correct rank $r = 70$. More importantly, for all these algorithms, their

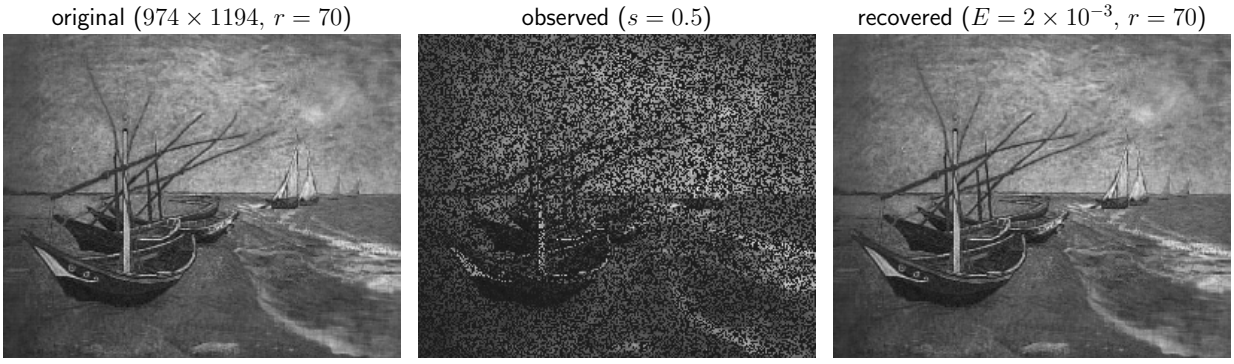

Figure 3: Matrix completion (77) of a grayscale image. *Left:* ground truth. *Middle:* sampling uniformly 50% of the entries. *Right:* example of a recovered image from a single run of our algorithms with $\mu = 0.05$. All algorithms performed similarly but the associated accelerated variant was much faster than the base method (see Fig. 1).

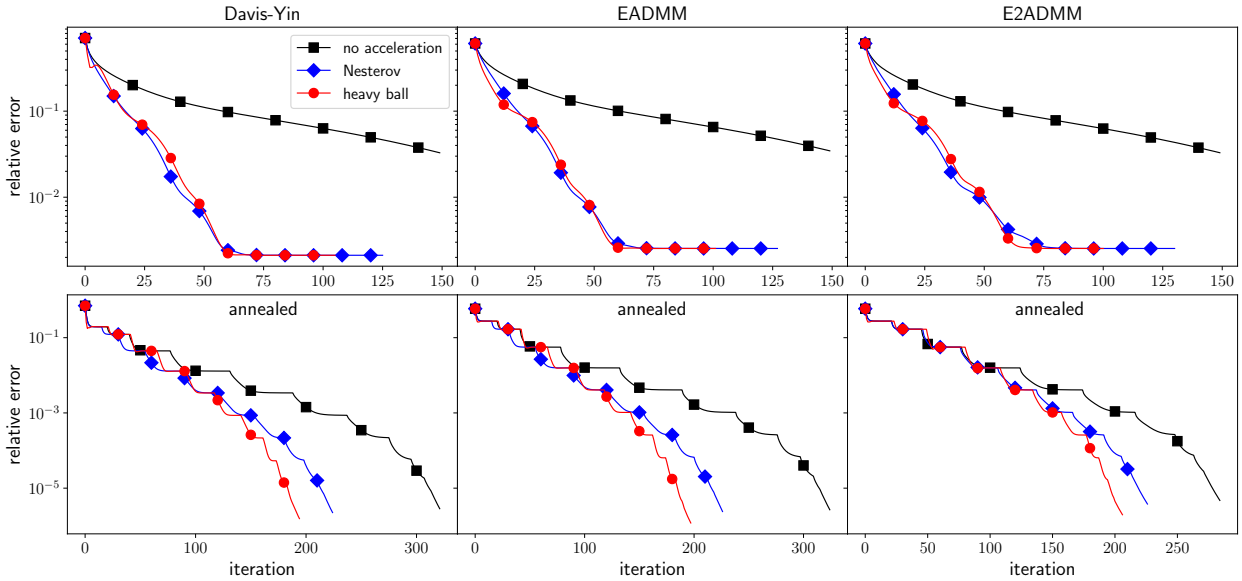

Figure 4: Convergence rates related to Fig. 3. *Top row:* single run of each algorithm with fixed $\mu = 0.05$. Only here we use $r_{\text{nest}} = 3$ and $r_{\text{hb}} = -\log 0.9$. The accelerated variants show significant faster convergence. *Bottom row:* same algorithms but with annealing on μ. We obtain much more accurate solutions. Importantly, the accelerated variants still outperform the base methods. (Davis-Yin=Algo 1/2; EADMM=Algo 5/6; E2ADMM=Algo 3/4.)

accelerated variants showed a significant faster convergence compared to the base method. The convergence rates with a single run and with an annealing schedule are shown in Fig. 4.

We now consider a harder case with a sampling rate $s = 0.3$, thus $d/p = 0.42$. The

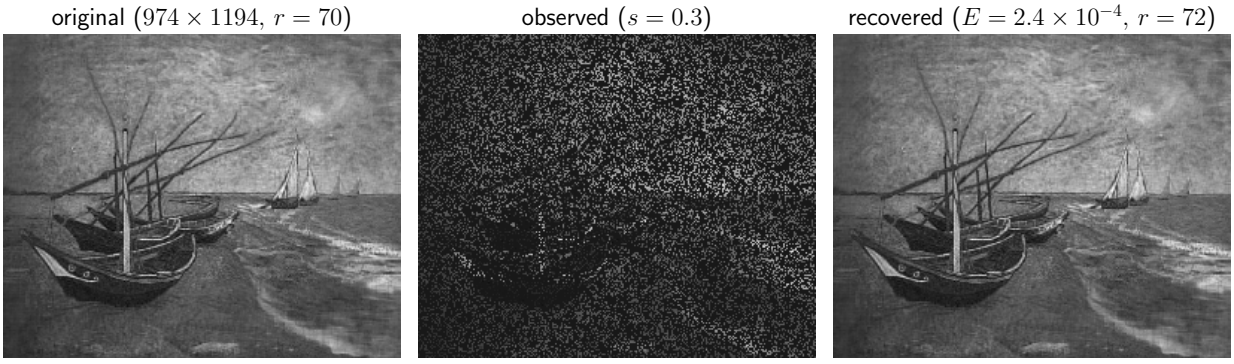

Figure 5: *Left:* ground truth. *Middle:* observed image with sampling rate $s = 0.3$. *Right:* recovered image under annealing and Nesterov accelerated variant (in this case heavy ball obtained relative error $\sim 10^{-3}$ but did not recover the rank). See convergence rates in Fig. 6.

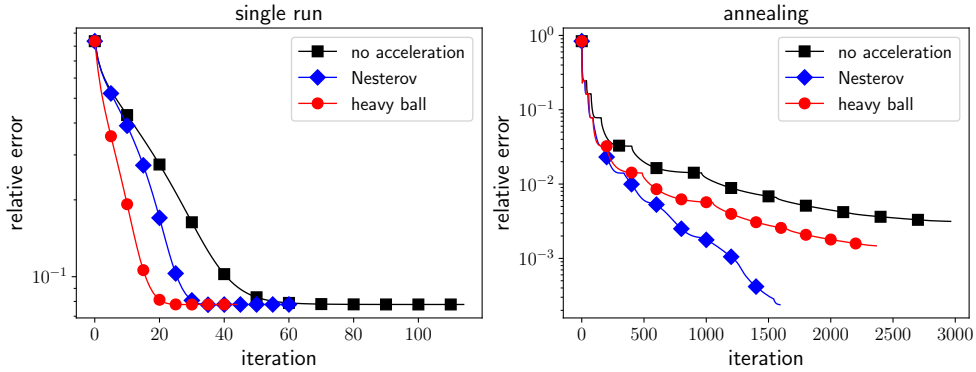

Figure 6: Convergence rates of single run *(Left)* and annealing *(Right)* for the problem of Fig. 5. We just show the results for Davis-Yin since similar results were obtained for both extensions of ADMM. Here the Nesterov accelerated variant had much better performance than heavy ball under annealing on μ.

observed image is shown in Fig. 5. When running the previous algorithms a single time with a fixed $\mu = 1$ they were able to recover the image with a relative error of 8×10^{-2} and obtain the correct rank $r = 70$. The accelerated variants improved the convergence (see left panel in Fig. 6). All methods performed closely but (accelerated) Davis-Yin was marginally better, thus we just show results for this case. Next, we use the same parameters but with the annealing schedule on μ (we choose $\bar{\mu} = 10^{-4}$). In this case the Nesterov accelerated variant outperformed heavy ball. It was able to recover the image with a relative error 2.4×10^{-4} and rank $r = 72$ (see right panel of Fig. 5) while heavy ball obtained a relative error of 1.4×10^{-3} and was unnable to reduce the rank (the nonaccelerated variants had worse relative error and was much slower). The convergence rates are shown in Fig. 6.

6 Conclusions

We have shown that all the four basic types of proximal optimization methods currently known, i.e. forward-backward, Tseng splitting, Douglas-Rachford and Davis-Yin, which are algorithms designed to converge to fixed point of nonexpansive operators, correspond to different discretizations of the gradient flow. We also showed that two types of accelerated variants of these methods arise naturally by employing a similar discretization approach to the accelerated gradient flow. All these methods preserve steady states of the underlying continuous dynamical system and are first order integrators. Moreover, we showed that ADMM, together with its accelerated variants, corresponds to a rebalanced splitting.

Our derivations provide a new perspective on the important class of operator splitting algorithms by establishing interesting connections with splitting methods of ODEs, which constitute a powerful approach for algorithm building. Noteworthy, the the gradient flow (1) and the accelerated gradient flow (7) have a unifying and central character for optimization, since numerous algorithms emerge from discretizations of these two dynamical systems.

We introduced several new algorithms, namely the accelerated variant of Davis-Yin (Algorithm 2), and two extensions of ADMM together with their respective accelerated variants (Algorithm 3 and 4, and Algorithms 5 and 6). The accelerated version of Tseng splitting (Algorithm 8) is also new to the best of our knowledge. We illustrated numerically that the accelerated variants of the base algorithms yield a faster convergence. We considered two settings, one for the plain algorithms and another with an annealing schedule.

Our results focused, on a first stage, in establishing interesting relationships between optimization and splitting methods for ODEs, which can form the basis for the construction of new accelerated methods. For future work, it would be important to obtain convergence rates for the proposed accelerated methods and stability analysis as well.

Acknowledgements

GF 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] 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.

[2] 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.

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

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

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

[6] C. J. Maddison, D. Paulin, Y. W. Teh, B. O'Donoghue, and A. Doucet, "Hamiltonian Descent Methods." arXiv:1809.05042 [math.OC], 2018.

[7] B. O'Donoghue and C. J. Maddison, "Hamiltonian Descent for Composite Objectives." arXiv:1806.02608 [math.OC], 2019.

[8] M. Muehlebach and M. I. Jordan, "A Dynamical Systems Perspective on Nesterov Acceleration," *ICML* (2019) . arXiv:1905.07436 [math.OC].

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

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

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

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

[13] 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.

[14] 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.

[15] G. França, D. P. Robinson, and R. Vidal, "ADMM and Accelerated ADMM as Continuous Dynamical Systems," *International Conference on Machine Learning* (2018) . arXiv:1805.06579 [math.OC].

[16] G. França, D. P. Robinson, and R. Vidal, "A Dynamical Systems Perspective on Nonsmooth Constrained Optimization." arXiv:1808.04048 [math.OC], 2018.

[17] 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.

[18] 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.

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

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

[21] 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.

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

[23] 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.

[24] 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.

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

[26] 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.

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

[28] S. MacNamara and G. Strang, "Operator Splitting." 2016.

[29] 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.

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

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

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

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

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

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

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

[37] 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.

[38] 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.

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

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

[41] 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.

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

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

[44] 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.

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

[46] 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.

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

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

[49] G. Strang, "On the Construction and Comparison of Difference Schemes," *SIAM J. Numer. Anal.* **5** no. 3, (1968) 506–517.

[50] 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.

[51] N. Parikh and S. Boyd, "Proximal Algorithms," *Found. Trends in Opt.* **1** no. 3, (2013) 123–231.

[52] 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.

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

[54] A. Beck and M. Teboulle, "A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems," *SIAM J. Imag. Sci.* **2** no. 1, (2009) 183–202.

[55] 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.

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

[57] 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.

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

[59] 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, 170–194.