JMLR: Workshop and Conference Proceedings vol 65:1–25, 2017 30th Annual Conference on Learning Theory

Integration Methods and Accelerated Optimization Algorithms

author names withheld

Editor: Under Review for COLT 2017

Abstract

We show that accelerated optimization methods can be seen as particular instances of multi-step integration schemes from numerical analysis, applied to the gradient flow equation. In comparison with recent advances in this vein, the differential equation considered here is the basic gradient flow and we show that multi-step schemes allow integration of this differential equation using larger step sizes, thus intuitively explaining acceleration results.

Introduction

The gradient descent algorithm used to minimize a function f has a well-known simple numerical interpretation as the integration of the gradient flow equation, written

$$\begin{aligned} x(0) &= x_0 \\ \dot{x}(t) &= -\nabla f(x(t)), \end{aligned} \tag{Gradient Flow}$$

using Euler's method. This appears to be a somewhat unique connection between optimization and numerical methods, since these two fields have inherently different goals. On one hand, numerical methods aim to get a precise discrete approximation of the solution $x(t)$ on a finite time interval. More sophisticated methods than Euler's were developed to get better consistency with the continuous time solution but still focus on a finite time horizon (see for example Süli and Mayers, 2003). On the other hand, optimization algorithms seek to find the minimizer of a function, which corresponds to the infinite time horizon of the gradient flow equation. Structural assumptions on f led to more sophisticated algorithms than the gradient method, such as the mirror gradient method (see for example Ben-Tal and Nemirovski, 2001; Beck and Teboulle, 2003), proximal gradient method (Nesterov et al., 2007) or a combination thereof (Duchi et al., 2010; Nesterov, 2015). Among them Nesterov's accelerated gradient algorithm (Nesterov, 1983) is proven to be optimal on the class of smooth convex or strongly convex functions. This last method was designed with the lower complexity bounds in mind, but the proof relies on purely algebraic arguments and the key mechanism behind acceleration remains elusive, which led to various interpretations of it (Bubeck et al., 2015; Allen Zhu and Orecchia, 2017; Lessard et al., 2016).

A recent stream of papers recently used differential equations to model the acceleration behavior and offer a better interpretation of Nesterov's algorithm (Su et al., 2014; Krichene et al., 2015; Wibisono et al., 2016; Wilson et al., 2016). However, the differential equation is often quite complex, being reverse-engineered from Nesterov's method itself, thus losing the intuition. Moreover, integration methods for these differential equations are often ignored or are not derived from standard numerical integration schemes, because the convergence proof of the algorithm does not require the continuous time interpretation.

Here, we take another approach. Rather than using a complicated differential equation, we use advanced multi-step methods to discretize the basic gradient flow equation in (Gradient Flow). These lesser known methods, developed decades ago by numerical analysts, directly correspond to various well-known optimization algorithms. In particular, Nesterov's method can be seen as a stable and consistent gradient flow discretization scheme that allows bigger step sizes in integration, leading to faster convergence.

The paper is organized as follows. In Section 1 we present our setting and recall classical results on differential equations. We then review definitions and theoretical properties of integration methods called linear multi-step methods in Section 2. Linear one-step and two-step methods are detailed in Section 3 and linked to optimization algorithms in Section 4 (strongly convex case) and Section 5 (convex case).

1. Gradient flow

We seek to minimize a L-smooth μ-strongly convex function defined on $\mathbb{R}^d$. We discretize the gradient flow equation (Gradient Flow), given by the following ordinary differential equation (ODE)

$$\begin{aligned} \dot{x}(t) &= g(x(t)) \\ x(0) &= x_0, \end{aligned} \tag{ODE}$$

where g comes from a potential $-f$, meaning $g = -\nabla f$. Smoothness of f means Lipschitz continuity of g, i.e.

$$\|g(x) - g(y)\| \leq L\|x - y\|, \quad \text{for every } x, y \in \mathbb{R}^d,$$

where $\|.\|$ is the Euclidean norm. This property ensures the existence and uniqueness of the solution of (ODE) (see Theorem E.2 in Appendix). Strong convexity of f also means strong monotonicity of $-g$, i.e.,

$$\mu\|x - y\|^2 \leq -\langle x - y, g(x) - g(y)\rangle,$$

and ensures that (ODE) has a unique point x^* such that $g(x^*) = 0$, called the equilibrium. This is the minimizer of f and the limit point of the solution, i.e., $x(\infty) = x^*$. In the numerical analysis literature, strong monotonicity of $-g$ is referred to as one-sided Lipschitz-continuity of g. Whatever its name, this property essentially contracts solutions of (ODE) with different initial points as detailed in Appendix E.2. Finally this assumption allows us to control the convergence rate of the potential f and the solution $x(t)$ as recalled in the following proposition.

Proposition 1.1 *Let f be a L-smooth and μ-strongly convex function and $x_0 \in \mathbf{dom}(f)$. Writing x^* the minimizer of f, the solution $x(t)$ of* (Gradient Flow) *satisfies*

$$f(x(t)) - f(x^*) \leq (f(x_0) - f(x^*))e^{-2\mu t} \tag{1}$$

$$\|x(t) - x^*\| \leq \|x_0 - x^*\|e^{-\mu t}. \tag{2}$$

A proof of this last result can be found in Appendix E.3 and we now focus on numerical methods to integrate (ODE).

2. Numerical integration of differential equations

In general, we do not have access to an explicit solution $x(t)$ of (ODE). We thus use integration algorithms to approximate the curve $(t, x(t))$ by a grid $(t_k, x_k) \approx (t_k, x(t_k))$ on a finite interval $[0, t_{\max}]$. For simplicity here, we assume the step size $h_k = t_k - t_{k-1}$ is constant, i.e., $h_k = h$ and $t_k = kh$. The goal is then to minimize the approximation error $\|x_k - x(t_k)\|$ for $k \in [0, t_{\max}/h]$. We first introduce Euler's method to illustrate this on a basic example.

2.1. Euler's explicit method

Euler's (explicit) method is one of the oldest and simplest schemes for integrating the curve $x(t)$. The idea stems form the Taylor expansion of $x(t)$ which reads

$$x(t+h) = x(t) + h\dot{x}(t) + O(h^2).$$

When $t = kh$, Euler's method approximates $x(t+h)$ by x_{k+1} by neglecting the second order term,

$$x_{k+1} = x_k + hg(x_k).$$

In optimization terms, we recognize the gradient descent algorithm used to minimize f. Approximation errors in an integration method accumulate with iterations, and as Euler's method uses only the last point to compute the next one, it has only limited control over the accumulated error.

2.2. Linear multi-step methods

Multi-step methods use a combination of several past iterates to improve convergence. Throughout the paper, we focus on *linear* s-step methods whose recurrence can be written

$$x_{k+s} = -\sum_{i=0}^{s-1} \rho_i x_{k+i} + h \sum_{i=0}^{s} \sigma_i g(x_{k+i}), \quad \text{for } k \geq 0, \tag{3}$$

where $\rho_i, \sigma_i \in \mathbb{R}$ are the parameters of the multi-step method and h is again the step size. Each new point x_{k+s} is a function of the information given by the s previous points. If $\sigma_s = 0$, each new point is given explicitly by the s previous points and the method is then called **explicit**. Otherwise each new point requires solving an implicit equation and the method is then called **implicit**.

To simplify notations we use the shift operator E, which maps $Ex_k \to x_{k+1}$. Moreover, if we write $g_k = g(x_k)$, then the shift operator also maps $Eg_k \to g_{k+1}$. Recall that a univariate polynomial is called monic if its leading coefficient is equal to 1. We now give the following concise definition of s-step linear methods.

Definition 2.1 *Given an* (ODE) *defined by* g, x_0*, a step size* h *and* $x_1, \ldots, x_{s-1}$ *initial points, a* ***linear*** s***-step method*** *generates a sequence* (t_k, x_k) *which satisfies*

$$\rho(E)x_k = h\sigma(E)g_k, \quad \textit{for every } k \geq 0,$$

where ρ *is a monic polynomial of degree* s *with coefficients* ρ_i*, and* σ *a polynomial of degree* s *with coefficients* σ_i*.*

A linear $s-$step method is uniquely defined by the polynomials (ρ, σ). The sequence generated by the method then depends on the initial points and the step size. Each linear multi step method has a twin sister, called *one-leg* method, which generates the sequence $\tilde{x}_k$ following

$$\rho(E)\tilde{x}_k = \sigma(1)hg\left(\frac{\sigma(E)}{\sigma(1)}\tilde{x}_k\right).$$

It is possible to show a bijection between x_k and $\tilde{x}_k$ (Dahlquist, 1983). We quickly mention one-leg methods in Appendix A but we will not go into more details here. We now recall a few results describing the performance of multi-step methods.

2.3. Stability

Stability is a key concept for integration methods. First of all, consider two curves $x(t)$ and $y(t)$, both solutions of (ODE), but starting from different points $x(0)$ and $y(0)$. If the function g is Lipchitz-continuous, it is possible to show that the distance between $x(t)$ and $y(t)$ is bounded on a finite interval, i.e.

$$\|x(t) - y(t)\| \leq C\|x(0) - y(0)\| \qquad \forall t \in [0, t_{\max}],$$

where C may depend exponentially on $t_{\max}$. We would like to have a similar behavior for our sequences x_k and y_k, approximating $x(t_k)$ and $y(t_k)$, i.e.

$$\|x_k - y_k\| \approx \|x(t_k) - y(t_k)\| \leq C\|x(0) - y(0)\| \qquad \forall k \in [0, t_{\max}/h], \tag{4}$$

when $h \to 0$, so $k \to \infty$. Two issues quickly arise, namely:

- For a linear s-step method, we need s starting values $x_0, ..., x_{s-1}$. Condition (4) will therefore depend on all these starting values and not only x_0.
- Any discretization scheme introduce at each step an approximation error, called local error, which is accumulated over time. We denote this error by $\epsilon^{\text{loc}}(x_{k+s})$ and define it as

  $$\epsilon^{\text{loc}}(x_{k+s}) \triangleq x_{k+s} - x(t_{k+s})$$

  if x_{k+s} is computed using the real solution $x(t_k), ..., x(t_{k+s-1})$.

In other words, the difference between x_k and y_k can be described as follows

$$\|x_k - y_k\| \ \leq \ \text{Error in the initial condition} + \text{Accumulation of local errors.}$$

We now write a complete definition of stability, inspired from Definition 6.3.1 from Gautschi (2011).

Definition 2.2 *A linear multi-step method is stable if, for two sequences x_k, y_k generated by (ρ, σ) using any sufficiently small step size $h > 0$, from the starting values $x_0, ..., x_{s-1}$, and $y_0, ..., y_{s-1}$, we have*

$$\|x_k - y_k\| \leq C\left(\max_{i\in\{0,...,s-1\}} \|x_i - y_i\| + \sum_{i=1}^{t_{\max}/h} \|\epsilon^{loc}(x_{i+s}) - \epsilon^{loc}(y_{i+s})\|\right), \tag{5}$$

for any $k \in [0, t_{\max}/h]$. Here, the constant C may depend on $t_{\max}$ but is independent of h.

When h tends to zero, we may recover equation (4) only if the accumulated local error tends also to zero. We thus need

$$\lim_{h \to 0} \frac{1}{h} \|\epsilon^{\text{loc}}(x_{i+s}) - \epsilon^{\text{loc}}(y_{i+s})\| = 0 \quad \forall i \in [0, t_{\max}/h].$$

This condition is called *consistency*. Once this condition satisfied, we still need to ensure

$$\|x_k - y_k\| \leq C \max_{i \in \{0,\dots,s-1\}} \|x_i - y_i\|, \tag{6}$$

and this condition is called *zero-stability*.

2.3.1. TRUNCATION ERROR AND CONSISTENCY

The truncation error of a linear multi-step method is a measure of the local error $\epsilon^{\text{loc}}(x_k)$ made by the method, normalized by h. More precisely it is defined using the difference between the step performed by the algorithm and the step which reaches exactly $x(t_{k+s})$, with

$$T(h) \triangleq \frac{x(t_{k+s}) - x_{k+s}}{h} \qquad \text{assuming } x_{k+i} = x(t_{k+i}),\ i = 0, \dots, s-1. \tag{7}$$

This definition does not depend on k but on the recurrence of the linear s-step method and on the (ODE) defined by g and x_0. We can use this truncation error to define consistency.

Definition 2.3 *An integration method for an* (ODE) *defined by g, x_0 is consistent if and only if, for any initial condition $x(t_0)$,*

$$\lim_{h \to 0} \|T(h)\| = 0.$$

The following proposition shows there exist simple conditions to check consistency, which rely on comparing a Taylor expansion of the solution with the coefficients of the method.

Proposition 2.4 *A linear multi step method defined by polynomials (ρ, σ) is consistent if and only if*

$$\rho(1) = 0 \qquad \textit{and} \qquad \rho'(1) = \sigma(1). \tag{8}$$

The proof is recalled in Appendix E.4. Consistency is crucial for integration methods, and some intuition is given in Appendix D.

2.3.2. ZERO-STABILITY AND ROOT CONDITION

To get stability, assuming consistency holds as above, we also need to satisfy the zero-stability condition (6), which characterizes the sensitivity of a method to initial conditions. Actually, the name comes from an interesting fact. Analyzing the special case where $g = 0$ is completely equivalent to the general case, as stated in the *root condition theorem*. The analysis becomes simplified, and reduces to standard linear algebra results because we only need to look at the solution of the homogeneous difference equation $\rho(E)x_k = 0$.

Theorem 2.5 (Root condition) *Consider a linear multi-step method (ρ, σ). The method is zero-stable if and only if all roots of $\rho(z)$ are in the unit disk, and the roots on the unit circle are simple.*

The proof of this theorem is technical and can be found as Theorem 6.3.4 of Gautschi (2011).

2.4. Convergence of the global error and Dahlquist's equivalence theorem

Numerical analysis focuses on integrating an ODE on a finite interval of time $[0, t_{\max}]$. It studies the behavior of the global error defined by the difference between the scheme and the solution of the ODE, i.e., $x(t_k) - x_k$, as a function of the step size h. If the global error converges to 0 with the step size, the method is guaranteed to approximate correctly the ODE on the time interval, for h small enough. The faster it converges with h, the less points we need to guarantee a global error within a given accuracy.

We now state *Dahlquist's equivalence theorem*, which shows that the global error converges to zero when h does if the method is *stable*, i.e., when the method is *consistent* and *zero-stable*. This naturally needs the additional assumption that the starting values $x_0, \dots, x_{s-1}$ are computed such that they converge to the solution $(x(0), \dots, x(t_{s-1}))$.

Theorem 2.6 (Dahlquist's equivalence theorem (simplified)) *Given an* (ODE) *defined by g and x_0 and a consistent linear multi-step method (ρ, σ), whose starting values are computed such that $\lim_{h\to 0} x_i = x(t_i)$ for any $i \in \{0, \dots, s-1\}$, zero-stability is necessary and sufficient for being convergent, i.e., $x(t_k) - x_k$ tends to zero for any k when the step size h tends to zero.*

Again, the proof of the theorem can be obtained from Gautschi (2011). Notice that this theorem is fundamental in numerical analysis. For example, it says that if a *consistent* method is not zero-stable, then the global error may be arbitrary large when the step size goes to zero, *even if the local error decreases*. In fact, if zero-stability is not satisfied, there exists a sequence generated by the linear multi-step method which grows with arbitrarily large factors.

2.5. Region of absolute stability

Stability and global error are ensured on finite time intervals, however solving optimization problems requires us to look at the infinite time horizon. We thus need more refined results and start by finding conditions ensuring that the numerical solution does not diverge when the time interval increases, i.e. that the numerical solution is stable with a constant C which *does not depend of* $t_{\max}$. Formally, for a fixed step-size h, we want to ensure

$$\|x_k\| \leq C \max_{i\in\{0,\dots,s-1\}} \|x_i\| \quad \text{for all } k \in [0, t_{\max}/h], \text{ and } t_{\max} > 0. \tag{9}$$

This is not possible without assumptions on the function g as in the general case the solution $x(t)$ itself may diverge. We begin with the simple scalar linear case which, given $\lambda > 0$, reads

$$\begin{aligned} \dot{x}(t) &= -\lambda x(t) \\ x(0) &= x_0. \end{aligned} \tag{Scalar Linear ODE}$$

The recurrence of a linear multi-step methods with parameters (ρ, σ) applied to (Scalar Linear ODE) then reads

$$\rho(E)x_k = -\lambda h\sigma(E)x_k \quad \Leftrightarrow \quad [\rho + \lambda h\sigma](E)x_k = 0,$$

where we recognize an homogeneous recurrence equation. Condition (9) is then controlled by the step size h and the constant λ, ensuring that this homogeneous recurrent equation produces bounded solutions. This leads us to the definition of the region of absolute stability.

Definition 2.7 *The region of absolute stability of a linear multi-step method defined by (ρ, σ) is the set of values λh such that the characteristic polynomial*

$$\pi_{\lambda h} \triangleq \rho + \lambda h \sigma \tag{10}$$

of the homogeneous recurrent equation $\pi_{\lambda h}(E)x_k = 0$ produces bounded solutions.

Standard linear algebra links this condition to the roots of the characteristic polynomial as recalled in the next proposition (see Lemma 12.1 of Süli and Mayers (2003)).

Proposition 2.8 *Let π be a polynomial and write x_k a solution of the homogeneous recurrent equation $\pi(E)x_k = 0$ with arbitrary initial values. If all roots of π are inside the unit disk and the ones on the unit circle have a multiplicity exactly equal to one, then $\|x_k\| \leq \infty$.*

Absolute stability of a linear multi-step method determines its ability to integrate a linear ODE defined by

$$\begin{aligned} \dot{x}(t) &= -Ax(t) \\ x(0) &= x_0, \end{aligned} \tag{Linear ODE}$$

where A is a positive symmetric matrix whose eigenvalues belong to $[\mu, L]$ for $0 < \mu \leq L$. In this case the step size h must indeed be chosen such that for any $\lambda \in [\mu, L]$, λh belongs to the region of absolute stability of the method. This (Linear ODE) is a special instance of (Gradient Flow) where f is a quadratic function. Therefore absolute stability gives necessary (but not sufficient) condition to integrate (Gradient Flow) on L-smooth μ-strongly convex functions.

2.6. Convergence analysis in the linear case

By construction, absolute stability also gives us hints on the convergence of x_k to the equilibrium. More precisiely, it allows us to control the rate of convergence of x_k, approximating the solution $x(t)$ of (Linear ODE).

Proposition 2.9 *Given a (Linear ODE) defined by x_0 and a positive symmetric matrix A whose eigenvalues belong to $[\mu, L]$ for $0 < \mu \leq L$, for a fixed step size h and a linear multi-step method defined by (ρ, σ), let $r_{\max}$ be defined as*

$$r_{\max} = \max_{\lambda \in [\mu, L]} \max_{r \in \mathrm{roots}(\pi_{\lambda h}(z))} |r|,$$

where $\pi_{\lambda h}$ is defined in (10). If $r_{\max} < 1$ and its multiplicity is equal to m, then the speed of convergence of the sequence x_k produced by the linear multi-step method to the equilibrium x^* of the differential equation is given by

$$\|x_k - x^*\| = O(k^{m-1} r_{\max}^k). \tag{11}$$

We can now use these properties to analyze and design multi-step methods.

3. Analysis and design of multi-step methods

As shown before, we want to integrate (Gradient Flow) and Proposition 1.1 gives us a rate of convergence in the continuous case. If the method tracks $x(t)$ with sufficient accuracy, then the rate of the method will be close to the rate of convergence of $x(kh)$. So, *larger values of h yield faster convergence of $x(t)$ to the equilibrium x^**. However h cannot be too large, as the method may be too inaccurate and/or unstable as h increases. *Convergence rates of optimization algorithms are thus controlled by our ability to discretize the gradient flow equation using large step sizes.* We recall the different conditions that proper linear multi-step methods should follow.

- *Monic polynomial (Section 2.2).* This is a convention, otherwise dividing both sides of the difference equation of the multi-step method by ρ_s does not change the method.
- *Explicit method (Section 2.2).* We assume that the scheme is explicit in order to avoid solving a non-linear system at each step (Appendix A shows that implicit methods are linked to proximal methods).
- *Consistency (Section 2.3.1).* If the method is not consistent, then the local error does not converge when the step size goes to zero.
- *Zero-stability (Section 2.3.2).* Zero-stability ensures convergence of the global error (Section 2.4) when the method is also consistent.
- *Region of absolute stability (Section 2.5).* If λh is not inside the region of absolute stability for any $\lambda \in [\mu, L]$, then the method is divergent when $t_{\max}$ increases.

Using the remaining degrees of freedom, we will tune the algorithm to have the best rate of convergence on (Linear ODE), which corresponds to the optimization of a quadratic function. Indeed, as showed in Proposition 2.9, the largest root of $\pi_{\lambda h}(z)$ gives us the rate of convergence on quadratic functions (when $\lambda \in [\mu, L]$). Since smooth and strongly convex functions are close to be quadratic (they are in fact sandwiched between two quadratics), this will also give us a good idea of the rate of convergence on these functions.

We do not derive a proof of convergence of the sequence for a general smooth and (strongly) convex function (in fact, it is already proved by Nesterov (2013) or using Lyapunov techniques by Wilson et al. (2016)). But our results provide intuition on *why* accelerated methods converge faster.

3.1. Analysis and design of explicit Euler's method ($s = 1$)

In Section 2.1 we introduced Euler's method. In fact, we can view it as an explicit linear "multi-step" method with $s = 1$ defined by the polynomials

$$\rho(z) = -1 + z, \quad \sigma(z) = 1.$$

We can check easily that it is consistent (using Proposition 2.4) and zero-stable since $\rho(z)$ has only one root which lies on the unit circle (Theorems 2.5 and 2.6). We need to determine the region of absolute stability in order to have an idea about the maximum value that $h > 0$ can take before the method becomes unstable. Assume we want to integrate any μ-strongly convex and L-smooth

function f, with $0 \leq \mu < L$ with any starting value x_0. Then, we need to find the set of value of h such that the roots of the polynomial

$$\pi_{\lambda h}(z) = [\rho + \lambda h \sigma](z) = -1 + \lambda h + z, \quad \lambda \in [\mu, L]$$

are small. The unique root is $1 - \lambda h$ and we need to solve the following minimax problem

$$\min_h \max_{\lambda \in [\mu, L]} |1 - \lambda h| ,$$

in the variable $h > 0$. The solution of this optimization problem is $h^* = \frac{2}{L+\mu}$, its optimal value is $(L - \mu)/(L + \mu)$ and its rate of convergence is then

$$\|x_k - x^*\| = O\left(\left(\frac{1 - \mu/L}{1 + \mu/L} \right)^k \right).$$

We recover the optimal step size and the rate of convergence of the gradient method for a general smooth and strongly convex function (Nesterov, 2013).

3.2. Analysis of two-step methods ($s = 2$)

We will now analyze two-step methods. First we write the conditions of a good linear multi step method, introduced at the beginning of this section, into constraints on the coefficients.

$$\begin{aligned}
\rho_2 &= 1 && \text{(Monic polynomial)} \\
\sigma_2 &= 0 && \text{(Explicit method)} \\
\rho_0 + \rho_1 + \rho_2 &= 0 && \text{(Consistency)} \\
\sigma_0 + \sigma_1 + \sigma_2 &= \rho_1 + 2\rho_2 && \text{(Consistency)} \\
|\text{Roots}(\rho)| &\leq 1 && \text{(Zero-stability).}
\end{aligned}$$

If we use all equalities, we finally have three linear constraints, defined by the set $\mathcal{L}$:

$$\mathcal{L} = \begin{cases} \rho_1 &= -(1 + \rho_0), \\ \sigma_1 &= 1 - \rho_0 - \sigma_0, \\ |\rho_0| &< 1. \end{cases} \tag{12}$$

We will now try to find some condition on the remaining parameters in order to have a stable method. At first, let us analyze a condition on the roots of second order equations. Absolute stability requires that all roots of the polynomial $\pi_{\lambda h}$ are inside the unit circle. The following proposition gives us the values of the roots of $\pi_{\lambda h}$ as a function of the parameters ρ_i and σ_i.

Proposition 3.1 *Given constants $0 < \mu \leq L$, a step size $h > 0$ and a linear two-step method defined by (ρ, σ), under the conditions*

$$\begin{aligned}
(\rho_1 + \mu h \sigma_1)^2 &\leq 4(\rho_0 + \mu h \sigma_0), \\
(\rho_1 + L h \sigma_1)^2 &\leq 4(\rho_0 + L h \sigma_0),
\end{aligned}$$

the roots $r_\pm(\lambda)$ of $\pi_{\lambda h}$, defined in (10), are complex conjugate for any $\lambda \in [\mu, L]$. Moreover, the largest modulus root is equal to

$$\max_{\lambda \in [\mu, L]} |r_\pm(\lambda)|^2 = \max \left\{ \rho_0 + \mu h \sigma_0,\ \rho_0 + L h \sigma_0 \right\}. \tag{13}$$

The proof relies on roots of a second order equation and can be found in Appendix E.5. The next step is to minimize the largest modulus (13) in the coefficients ρ_i and σ_i to get the best rate of convergence, assuming the roots are complex. We will not develop the case were the roots are real because this leads to weaker results.

3.3. Design of optimal two-step method for quadratics

We have now have all ingredients to build a two-step method for which the sequence x_k converges quickly to x^* for quadratic functions. We need to solve the following problem,

$$\begin{array}{ll} \min & \max\{\rho_0 + \mu h\sigma_0,\ \rho_0 + Lh\sigma_0\} \\ \text{s.t.} & (\rho_0, \rho_1, \sigma_1) \in \mathcal{L} \\ & (\rho_1 + \mu h\sigma_1)^2 \leq 4(\rho_0 + \mu h\sigma_0) \\ & (\rho_1 + Lh\sigma_1)^2 \leq 4(\rho_0 + Lh\sigma_0), \end{array}$$

in the variables $\rho_0, \rho_1, \sigma_0, \sigma_1, h > 0$. where $\mathcal{L}$ is defined in (12). If we use the equality constraints in (12) and make the following change of variables,

$$\begin{cases} \hat{h} & = h(1-\rho_0), \\ c_\mu & = \rho_0 + \mu h\sigma_0, \\ c_L & = \rho_0 + Lh\sigma_0, \end{cases} \tag{14}$$

the problem becomes, for fixed $\hat{h}$,

$$\begin{array}{ll} \min & \max\{c_\mu,\ c_L\} \\ \text{s.t.} & (-1 - c_\mu + \mu\hat{h})^2 \leq 4c_\mu \\ & (-1 - c_L + L\hat{h})^2 \leq 4c_L \\ & |Lc_\mu - \mu c_L| < |L - \mu|, \end{array}$$

in the variables c_μ, c_L. In that case, the optimal solution is given by

$$c_\mu^* = \left(1 - \sqrt{\mu\hat{h}}\right)^2, \quad c_L^* = \left(1 - \sqrt{L\hat{h}}\right)^2, \tag{15}$$

obtained by tightening the two first inequalities, for $\hat{h} \in]0, \frac{(1+\mu/L)^2}{L}[$. Now if we fix $\hat{h}$ we can recover an optimal two step linear method defined by (ρ, σ) and an optimal step size h by using the equations in (14). We will use the following quantity

$$\beta \triangleq \frac{1 - \sqrt{\mu/L}}{1 + \sqrt{\mu/L}}. \tag{16}$$

A suboptimal two-step method. We can fix $\hat{h} = 1/L$ for example. All computations done, the parameters of this two-step method, called method $\mathcal{M}_1$, are

$$\mathcal{M}_1 = \begin{cases} \rho(z) & = \beta - (1+\beta)z + z^2, \\ \sigma(z) & = -\beta(1-\beta) + (1-\beta^2)z, \\ h & = \frac{1}{L(1-\beta)}, \end{cases} \tag{17}$$

and its largest modulus root (13) is given by

$$\text{rate}(\mathcal{M}_1) = \sqrt{\max\{c_\mu,\ c_L\}} = \sqrt{c_\mu} = 1 - \sqrt{\mu/L}.$$

Optimal two-step method for quadratics. We can compute the optimal $\hat{h}$ which minimizes the maximum of the two roots c_μ^* and c_L^* defined in (15). The solution is simply the one which balances the two terms in the maximum:

$$\hat{h}^* = \frac{(1+\beta)^2}{L} \quad \Rightarrow \quad c_\mu^* = c_L^*.$$

This choice of $\hat{h}$ leads to the method $\mathcal{M}_2$, described by

$$\mathcal{M}_2 = \begin{cases} \rho(z) & = \beta^2 - (1+\beta^2)z + z^2, \\ \sigma(z) & = (1-\beta^2)z, \\ h & = \frac{1}{\sqrt{\mu L}}, \end{cases} \tag{18}$$

with the rate of convergence

$$\text{rate}(\mathcal{M}_2) = \sqrt{c_\mu} = \sqrt{c_L} = \beta < \text{rate}(\mathcal{M}_1).$$

We will now see that methods $\mathcal{M}_1$ and $\mathcal{M}_2$ are actually related to Nesterov's method and Polyak's heavy ball algorithms.

4. On the link between integration and optimization

In the previous section, we derived a family of linear multi-step methods, parametrized by $\hat{h}$. We will now compare these methods to common optimization algorithms used to minimize L-smooth, μ-strongly convex functions.

4.1. Polyak's heavy ball method

The heavy ball method was proposed by Polyak (1964). It adds a momentum term to the gradient step

$$x_{k+2} = x_{k+1} - c_1 \nabla f(x_{k+1}) + c_2(x_{k+1} - x_k),$$

where $c_1 = 4/(\sqrt{L} + \sqrt{\mu})^2$ and $c_2 = \beta^2$ where β is defined in (16). We can organize the terms in the sequence to match the general structure of linear multi-step methods, to get

$$\beta^2 x_k - (1+\beta^2)x_{k+1} + x_{k+2} = c_1 \left(-\nabla f(x_{k+1})\right).$$

We easily identify $\rho(z) = \beta^2 - (1+\beta^2)z + z^2$ and $h\sigma(z) = c_1 z$. To extract h, we will assume that the method is consistent (see conditions (8)), which means

$$\begin{aligned} \rho(1) &= 0 && \text{Always satisfied} \\ h\rho'(1) &= h\sigma(1) && \Rightarrow \; h = \frac{c_1}{1-\beta^2} = \frac{1}{\sqrt{\mu L}}. \end{aligned}$$

All computations done, we can identify the "hidden" linear multi-step method as

$$\mathcal{M}_{\text{Polyak}} = \begin{cases} \rho(z) & = \beta^2 - (1+\beta^2)z + 1 \\ \sigma(z) & = (1-\beta^2)z \\ h & = \frac{1}{\sqrt{\mu L}}. \end{cases} \tag{19}$$

This shows that $\mathcal{M}_{\text{Polyak}} = \mathcal{M}_2$. In fact, this result was expected since Polyak's method is known to be optimal for quadratic functions. However, it is known that Polyak's algorithm does not converge for a general smooth and strongly convex function (Lessard et al., 2016).

4.2. Nesterov's accelerated gradient

Nesterov's accelerated method in its simplest form is described by two sequences x_k and y_k, with

$$\begin{aligned} y_{k+1} &= x_k - \frac{1}{L}\nabla f(x_k), \\ x_{k+1} &= y_{k+1} + \beta(y_{k+1} - y_k). \end{aligned}$$

As above, we will write Nesterov's accelerated gradient as a linear multi-step method by expanding y_k in the definition of x_k, to get

$$\beta x_k - (1+\beta)x_{k+1} + x_{k+2} = \frac{1}{L}\left(-\beta(-\nabla f(x_k)) + (1+\beta)(-\nabla f(x_{k+1}))\right).$$

Again, we will assume that the method is consistent, then identify the hidden linear multi-step method associated to Nesterov's algorithm. Indeed,

$$\begin{aligned} \rho(1) &= 0 && \text{Always satisfied} \\ h\rho'(1) &= h\sigma(1) && \Rightarrow\ h = \frac{1}{L(1-\beta)}. \end{aligned}$$

After identification,

$$\mathcal{M}_{\text{Nest}} = \begin{cases} \rho(z) &= \beta - (1+\beta)z + z^2, \\ \sigma(z) &= -\beta(1-\beta) + (1-\beta^2)z, \\ h &= \frac{1}{L(1-\beta)}, \end{cases}$$

which means that $\mathcal{M}_1 = \mathcal{M}_{\text{Nest}}$.

4.3. The convergence rate of Nesterov's method

Pushing the analysis a little bit further, we can show *why* Nesterov's algorithm is faster than gradient method. There is of course a complete proof of its rate of convergence (Nesterov, 2013), even using the argument of differential equations (Wibisono et al., 2016; Wilson et al., 2016), but we take a more intuitive approach here. The key parameter is the step size h. If we compare it with the one in classical gradient method, Nesterov's method uses a step size which is $(1-\beta)^{-1} \approx \sqrt{L/\mu}$ larger.

Recall that, in continuous time, we have seen the rate of convergence of $x(t)$ to x^* given by

$$f(x(t)) - f(x^*) \leq e^{-2\mu t}(f(x_0) - f(x^*)).$$

The gradient method tries to approximate $x(t)$ using Euler approximation with step size $h = 1/L$, which means $x_k^{(\text{grad})} \approx x(k/L)$, so

$$f(x_k^{(\text{grad})}) - f(x^*) \approx f(x(k/L)) - f(x^*) \leq (f(x_0) - f(x^*))e^{-2k\frac{\mu}{L}}.$$

However, Nesterov's method has the step size

$$h_{\text{Nest}} = \frac{1}{L(1-\beta)} = \frac{1+\sqrt{\mu/L}}{2\sqrt{\mu L}} \approx \frac{1}{\sqrt{4\mu L}} \qquad \text{which means} \quad x_k^{\text{nest}} \approx x\left(k/\sqrt{4\mu L}\right).$$

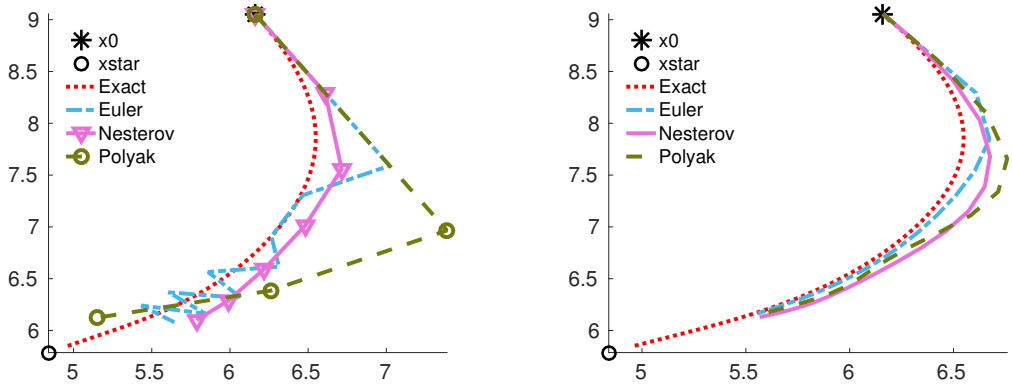

Figure 1: Integration of a linear ODE (which corresponds to the minimization of a quadratic function) using Euler's, Nesterov's and Polyak's method between $[0, t_{\max}]$. On the left, the optimal step size is used. Because Polyak's algorithm is the one with the biggest step size, it needs less iterations than Nesterov or Euler to approximate $x(t_{\max})$. On the right side, we reduced the step size to $1/L$ for all methods. We clearly observe they all track the same ODE: the gradient flow.

In that case, the estimated rate of convergence becomes

$$f(x_k^{\text{nest}}) - f(x^*) \approx f\left(x\left(k/\sqrt{4\mu L}\right)\right) - f(x^*) \leq (f(x_0) - f(x^*))e^{-k\sqrt{\mu/L}},$$

which is approximatively the rate of convergence of Nesterov's algorithm in discrete time and we recover the accelerated rate in $\sqrt{\mu/L}$ versus μ/L for gradient descent. The accelerated method is more efficient because it integrates the gradient flow *faster* than simple gradient descent, making longer steps. A numerical simulation in Figure 1 makes this argument more visual. This intuitive argument is still valid for the convex counterpart of Nesterov's accelerated gradient.

5. Acceleration for convex functions

By matching the coefficients of the Nesterov's method, we deduced the value of the step-size used for the integration of (Gradient Flow). Then, using the rate of convergence of $x(t)$ to x^*, we estimated the rate of convergence of Nesterov's method assuming $x_k \approx x(t_k)$. Here, we will do the same but without assuming strong convexity. However, the estimation of the rate of convergence in discrete time needs the one in continuous time, described by the following proposition.

Proposition 5.1 *Let f be L-smooth and convex and $x(t)$ be the solution of* (Gradient Flow)*. Then*

$$f(x(t)) - f(x^*) \leq \frac{\|x_0 - x^*\|^2}{t + (2/L)}. \tag{20}$$

The proof of this proposition can be found in E.6. Assume we use Euler's method with step size $h = \frac{1}{L}$, the estimated rate of convergence will be

$$f(x_k) - f(x^*) \approx f(x(kh)) - f(x^*) \leq \frac{L\|x_0 - x^*\|^2}{k+2},$$

which is close to the rate of convergence of the classical gradient method for convex function. Now, consider the Nesterov's method for minimizing a smooth and convex function f:

$$\begin{aligned} y_{k+1} &= x_k - \frac{1}{L}\nabla f(x_k) \\ x_{k+1} &= -\beta_k x_k + (1+\beta_k)x_{k+1}, \end{aligned}$$

where $\beta_k \approx \frac{k-2}{k+1}$. If we expand everything, we get after rearrangement,

$$\beta_k x_{k-1} - (1+\beta_k)x_k + x_{k+1} = \frac{1}{L}\left(\beta_k(-\nabla f(x_{k-1})) - (1+\beta_k)(-\nabla f(x_k))\right).$$

In other terms, we have an expression of the form $\rho_k(E)x_k = h_k\sigma_k(E)(-\nabla f(x_k))$. We can identify h if we assume the method consistent, which means

$$\begin{aligned} \rho(1) &= 0 && \text{Always satisfied} \\ h_k\rho_k'(1) &= h_k\sigma_k(1) && \Rightarrow\ h_k = \frac{1}{L(1-\beta_{k+1})} = \frac{(k+2)}{3L}. \end{aligned}$$

We can estimate, using (20), the rate of convergence of Nesterov's method. Since $x_k \approx x(t_k)$,

$$x_k \approx x\left(\sum_{i=0}^{k} h_i\right) \approx x\left(\frac{k^2}{6L}\right).$$

In terms of convergence to the optimal value,

$$f(x_k) - f(x^*) \approx f(x(t_k) - f(x^*) \leq \frac{6L\|x_0 - x^*\|^2}{k^2+12},$$

which is close to the bound from Nesterov (2013). Again, because the step-size of Nesterov's algorithm is larger (while keeping a stable sequence), we converge faster than the Euler's method.

6. Conclusion and future works

We connected several optimization algorithms to multi-step integration methods in numerical analysis. By using the theory of linear multi-step methods on the basic gradient flow equation, we recover Polyak's and Nesterov's method using some optimality arguments. This provides an intuitive interpretation for the design of these methods, with optimal step sizes giving a direct explanation of the acceleration phenomenon. Our approach generalizes to more structured problems by introducing the appropriate integration method and/or looking at a generalized gradient flow equation that takes into account the geometry of the problem. We illustrate the links between integration methods and proximal, mirror gradient descent and universal gradient methods in Appendices A, B and C respectively.

We described a simple interpretation of the acceleration phenomenon, but our analysis is still restricted to quadratic problems. The study of G-stability (Dahlquist, 1978; Butcher, 2006) may generalize our approach to smooth strongly convex functions. For the non-strongly convex case, the dependence in k of Nesterov's algorithm makes its links with integration methods less clear.

Finally the study of the links between optimization and numerical methods may provide new algorithms for both fields. Runge-Kutta methods (another way to integrate differentials equations) may lead to newer algorithms in optimization. Conversely, Nesterov's algorithm may lead to new integration methods to integrate the gradient flow equation of a convex function.

References

Zeyuan Allen Zhu and Lorenzo Orecchia. Linear coupling: An ultimate unification of gradient and mirror descent. In *Proceedings of the 8th Innovations in Theoretical Computer Science*, ITCS 17, 2017.

Uri M Ascher, Steven J Ruuth, and Brian TR Wetton. Implicit-explicit methods for time-dependent partial differential equations. *SIAM Journal on Numerical Analysis*, 32(3):797–823, 1995.

Amir Beck and Marc Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, 2003.

Aharon Ben-Tal and Arkadi Nemirovski. *Lectures on modern convex optimization: analysis, algorithms, and engineering applications*. SIAM, 2001.

Jérôme Bolte, Aris Daniilidis, and Adrian S. Lewis. The Łojasiewicz inequality for nonsmooth subanalytic functions with applications to subgradient dynamical systems. *SIAM Journal on Optimization*, 17(4): 1205–1223, 2007.

S. Bubeck, Y. Tat Lee, and M. Singh. A geometric alternative to nesterov's accelerated gradient descent. *ArXiv e-prints*, jun 2015.

JC Butcher. Thirty years of g-stability. *BIT Numerical Mathematics*, 46(3):479–489, 2006.

Germund Dahlquist. G-stability is equivalent toa-stability. *BIT Numerical Mathematics*, 18(4):384–401, 1978.

Germund Dahlquist. On one-leg multistep methods. *SIAM journal on numerical analysis*, 20(6):1130–1138, 1983.

John C Duchi, Shai Shalev-Shwartz, Yoram Singer, and Ambuj Tewari. Composite objective mirror descent. In *COLT*, pages 14–26, 2010.

Jason Frank, Willem Hundsdorfer, and JG Verwer. On the stability of implicit-explicit linear multistep methods. *Applied Numerical Mathematics*, 25(2-3):193–205, 1997.

Walter Gautschi. *Numerical analysis*. Springer Science & Business Media, 2011.

Walid Krichene, Alexandre Bayen, and Peter L Bartlett. Accelerated mirror descent in continuous and discrete time. In *Advances in neural information processing systems*, pages 2845–2853, 2015.

Laurent Lessard, Benjamin Recht, and Andrew Packard. Analysis and design of optimization algorithms via integral quadratic constraints. *SIAM Journal on Optimization*, 26(1):57–95, 2016.

Yurii Nesterov. A method of solving a convex programming problem with convergence rate o (1/k2). In *Soviet Mathematics Doklady*, volume 27, pages 372–376, 1983.

Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

Yurii Nesterov. Universal gradient methods for convex optimization problems. *Mathematical Programming*, 152(1-2):381–404, 2015.

Yurii Nesterov et al. Gradient methods for minimizing composite objective function, 2007.

Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

R Tyrrell Rockafellar. Monotone operators and the proximal point algorithm. *SIAM journal on control and optimization*, 14(5):877–898, 1976.

Weijie Su, Stephen Boyd, and Emmanuel Candes. A differential equation for modeling nesterovs accelerated gradient method: Theory and insights. In *Advances in Neural Information Processing Systems*, pages 2510–2518, 2014.

Endre Süli and David F Mayers. *An introduction to numerical analysis*. Cambridge university press, 2003.

Andre Wibisono, Ashia C Wilson, and Michael I Jordan. A variational perspective on accelerated methods in optimization. *Proceedings of the National Academy of Sciences*, page 201614734, 2016.

Ashia C Wilson, Benjamin Recht, and Michael I Jordan. A lyapunov analysis of momentum methods in optimization. *arXiv preprint arXiv:1611.02635*, 2016.

Gengen Zhang and Aiguo Xiao. Stability and convergence analysis of implicit–explicit one-leg methods for stiff delay differential equations. *International Journal of Computer Mathematics*, 93(11):1964–1983, 2016.

Appendix A. Proximal algorithms and implicit integration methods

We present here links between proximal algorithms and implicit numerical methods. that integrate the gradient flow equation. We begin with Euler's implicit method that corresponds to the proximal point algorithm.

A.1. Euler's implicit method and proximal point algorithm

We saw in Section 2.1 that Euler's explicit method used the Taylor expansion of the solution $x(t)$ of the (ODE) at the current point. The implicit version uses the Taylor expansion at the next point which reads

$$x(t) = x(t+h) - h\dot{x}(t+h) + O(h^2).$$

If $t = kh$, by neglecting the second order term we get implicit Euler's method,

$$x_{k+1} = x_k + hg(x_{k+1}). \tag{21}$$

This recurrent equation requires to solve an implicit equation at each step that may be costly. However it provides better stability than the explicit version. This is generally the case for implicit methods (see Süli and Mayers (2003) for further details on implicit methods).

Now assume that g comes from a potential $-f$ such that we are integrating (Gradient Flow). Solving the implicit equation (21) is equivalent to compute the proximal operator of f defined as

$$\mathbf{prox}_{f,h}(x) = \operatorname*{argmin}_z \frac{1}{2}\|z - x\|_2^2 + hf(z). \tag{22}$$

This can be easily verified by checking the first-order optimality conditions of the minimization problem. Euler's implicit method applied to (Gradient Flow) reads then

$$x_{k+1} = \mathbf{prox}_{f,h}(x_k),$$

where we recognize the proximal point algorithm (Rockafellar, 1976).

We present now Mixed ODE that corresponds to composite optimization problems.

A.2. Implicit Explicit methods and proximal gradient descent

In numerical analysis, it is common to consider the differential equation

$$\dot{x} = g(x) + \omega(x), \tag{Mixed ODE}$$

where $g(x)$ is considered as the "non-stiff" part of the problem and ω the stiff one, where stiffness may be assimilated to bad conditioning (Ascher et al., 1995; Frank et al., 1997).

Usually, ω is also assumed to be simple, i.e., it is possible to integrate ω using an implicit method. If ω derives from a potential $-\Omega$ (meaning $\omega = -\nabla\Omega$), this is equivalent to assume that the proximal operator of Ω defined in (22) can be computed in closed form or with cheap computations.

We approximate the solution of (Mixed ODE) using IMplicit-EXplicit schemes (IMEX). In our case, we will focus on the following multi-step based IMEX scheme,

$$\rho(E)x_k = h\big(\sigma(E)g(x_k) + \gamma(E)\omega(x_k)\big),$$

where ρ, σ and γ are polynomials of degrees s, $s-1$ (the explicit part) and s respectively and ρ is monic. It means that, at each iteration, we need to solve

$$x_{k+s} = \sum_{i=0}^{s-1} \underbrace{\left(-\rho_i x_{k+i} + \sigma_i h g(x_{k+i}) + \gamma_i h \omega(x_{k+i})\right)}_{\text{known}} + \gamma_s \omega(x_{k+s}),$$

in x_{k+s}.

In terms of optimization the mixed ODE corresponds to composite minimization problems of the form

$$\text{minimize} \quad f(x) + \Omega(x), \tag{23}$$

where f, g are convex and g has a computable proximal operator. We can link IMEX schemes with many optimization algorithms which use the proximal operator, such as proximal gradient method, FISTA or Nesterov's method. For example, proximal gradient is written

$$\begin{aligned} y_{k+1} &= x_k + h g(x_k) \\ x_{k+1} &= \mathbf{prox}_{h\Omega}(y_{k+1}). \end{aligned}$$

After expansion, we get

$$x_{k+1} = y_{k+1} + h\omega(x_{k+1}) = x_k + h g(x_k) + h\omega(x_{k+1}),$$

which corresponds to the IMEX method with polynomials

$$\rho(z) = -1 + z, \quad \sigma(z) = 1, \quad \gamma(z) = z.$$

However, for Fista and Nesterov's method, we need to use a variant of linear multi-step algorithms, called *one leg* methods (Dahlquist, 1983; Zhang and Xiao, 2016). Instead of combining the gradients, the idea is to compute g at a linear combination of the previous points, i.e.

$$\rho(E) x_k = h \left(g(\sigma(E) x_k) + \omega(\gamma(E) x_k)\right).$$

Their analysis (convergence, consistency, interpretation of h, etc...) is slightly different from linear multi-step method, so we will not go into details in this paper, but the correspondence still holds.

A.3. Non-smooth gradient flow

In the last subsection we assumed that ω comes from a potential. However in the optimization literature, composite problems have a smooth convex part and a non-smooth sub-differentiable convex part which prevents us from interpreting the problem with the gradient flow ODE. Non-smooth convex optimization problems can be treated with differential inclusions (Bolte et al., 2007)

$$\dot{x}(t) + \partial f(x(t)) \ni 0,$$

for f a sub-differentiable function whose sub-differential at x is written $\partial f(x)$. Composite problems (23) can then be seen as the discretization of the differential inclusion

$$\dot{x}(t) + \nabla f(x(t) + \partial \Omega x(t) \ni 0.$$

Appendix B. Mirror gradient descent and non-Euclidean gradient flow

In many optimization problems, it is common to replace the Euclidean geometry with a distance-generating function called $d(x)$, with the associated Bregman divergence

$$\mathcal{B}_d(x, y) = d(x) - d(y) - \langle \partial d(y), x - y \rangle,$$

with d strongly-convex and lower semi-continuous. To take into account this geometry we consider the Non-Euclidean Gradient Flow (Krichene et al., 2015)

$$\begin{aligned} \dot{y}(t) &= -\nabla f\left(x(t)\right) \\ x(t) &= \nabla d^*(y(t)) \\ x(0) &= x_0,\ y(0) = \nabla d(x_0). \end{aligned} \tag{NEGF}$$

Here ∇d maps primal variables to dual ones and, as d is strongly convex, $(\nabla d)^{-1} = \nabla d^*$, where d^* is the Fenchel conjugate of d. In fact, we can write (NEGF) using only one variable y, but this formulation has the advantage to exhibit both primal and dual variables $x(t)$ and $y(t)$. Applying the forward Euler's explicit method we get the following recurrent equation

$$y_{k+1} - y_k = -h\nabla f(x_k), \quad x_{k+1} = \nabla d^* y_{k+1}.$$

Now consider the mirror gradient scheme :

$$x_{k+1} = \operatorname*{argmin}_x \ h\langle \nabla f(x_k), x\rangle + \mathcal{B}_h(x, x_k).$$

First optimality condition reads

$$\nabla_x \left(h\langle \nabla f(x_k), x\rangle + \mathcal{B}_h(x, x_k)\right)\big|_{x=x_{k+1}} = h\nabla f(x_k) + \nabla d(x_{k+1}) - \nabla d(x_k) = 0$$

Using that $(\nabla d)^{-1} = \nabla d^*$ we get

$$h\nabla f(x_k) + y_{k+1} - y_k = 0, \quad x_{k+1} = \nabla d^* y_{k+1},$$

which is exactly Euler's explicit method defined in (NEGF).

Appendix C. Universal gradient descent and generalized gradient flow

Consider the Generalized Gradient Flow, which combines the ideas of (Mixed ODE) and (NEGF),

$$\begin{aligned} \dot{y}(t) &= -\nabla f(x(t)) - \nabla \Omega(x(t)) \\ x(t) &= \nabla d^*(y(t)) \\ x(0) &= x_0,\ y(0) = \nabla d(x_0). \end{aligned} \tag{GGF}$$

We can write its ODE counterpart, called the "Generalized ODE",

$$\begin{aligned} \dot{y}(t) &= g(x(t)) + \omega(x(t)) \\ x(t) &= \nabla d^*(y(t)), \\ x(0) &= x_0,\ y(0) = \nabla d(x_0). \end{aligned} \tag{GODE}$$

where $g = -\nabla f$, with f a smooth convex function, d a strongly convex and semi-continuous distance generating function and $\omega = -\nabla\Omega$, where Ω is a simple convex function. If Ω is not differentiable we can consider the corresponding differential inclusion as presented in Section A.3. Here we focus on (GODE) and (GGF) to highlight the links with integration methods. The discretization of this ODE is able to generate many algorithms in many different settings. For example, consider the IMEX explicit-implicit Euler's method,

$$\frac{y_{k+1} - y_k}{h} = g(x_k) + \omega(d^*(y_{k+1})), \quad x_{k+1} = \nabla d^*(y_{k+1}),$$

which can be decomposed into three steps,

$$\begin{aligned} z_{k+1} &= y_k + hg\left(x_k\right) && \text{(Gradient step in the dual space)}, \\ y_{k+1} &= \mathbf{prox}_{h(\Omega\circ\nabla d^*)}\left(z_{k+1}\right) && \text{(Projection step in the dual space)}, \\ x_{k+1} &= \nabla d^*(y_{k+1}) && \text{(Back in the primal space)}. \end{aligned} \tag{24}$$

Now consider the universal gradient method scheme presented by Nesterov (2015):

$$x_{k+1} = \arg\min_x \langle \nabla f(x_k), x - x_k\rangle + \Omega(x) + \mathcal{B}_d(x, x_k).$$

Again we can show that both recursions are the same: if we write the first optimality condition,

$$\begin{aligned} 0 &= \nabla_x \left(h\langle \nabla f(x_k), x - x_k\rangle + h\Omega(x) + \mathcal{B}(x, x_k)\right)\big|_{x=x_{k+1}} \\ &= hf(x_k) + h\partial\Omega(x_{k+1}) + \nabla d(x_{k+1}) - \nabla d(x_k) \\ &= \underbrace{hf(x_k) - y_k}_{=z_{k+1}} + h\partial\Omega(\nabla d^*(y_{k+1})) - y_{k+1}. \end{aligned}$$

We thus need to solve the non-linear system of equations

$$y_{k+1} = z_{k+1} + h\partial\Omega(\nabla d^*(y_{k+1})),$$

which is completely equivalent to the second step of (24). Then we simply recover x_{k+1} by applying ∇d^* on y_{k+1}.

Appendix D. Interpretation of consistency conditions

Proposition *A method is consistent if and only if*

$$\rho(1) = 0 \qquad \textit{and} \qquad \rho'(1) = \sigma(1).$$

First condition, $\rho(1) = 0$. If the condition is not satisfied, then the method exhibits an artificial gain or damping. Assume we start at some equilibrium x^* of ODE (i.e. $\nabla f(x^*) = 0$), so $x_i = x^*$ for the first $s - 1$ steps. The next iterate becomes

$$x_s = -\sum_{i=0}^{s-1} \rho_i x^* + h\sigma(E)\underbrace{g(x^*)}_{=0},$$

and if $1 + \sum_{i=0}^{s} \rho_i = \rho(1) \neq 0$, we have that the next iterate x_s is different from x^*.

Second condition, $\rho'(1) = \sigma(1)$. If this relation is not satisfied, we actually are integrating another equation than (ODE). Assuming the first condition satisfied is a root of ρ, consider the factorization

$$\rho(z) = (z-1)\tilde{\rho}(z) \qquad \Rightarrow \qquad \tilde{\rho}(1) = \rho'(1),$$

where $\tilde{\rho}$ is a polynomial of degree $s-1$, and $\rho'(1) = \tilde{\rho}(1)$. The linear multi-step method becomes

$$\tilde{\rho}(E)(y_{k+1} - y_k) = h\sigma(E)g(y_0).$$

Let $G_k = \sum_{i=1}^{k} hg(x_i)$. If we sum up the above equation from $k = 0$, the relation becomes

$$\tilde{\rho}(E)(y_k - y_0) = \sigma(E)G_k.$$

If h goes to zero, our iterates x_k converge to some continuous curve $c(t)$, and $G_k \to \int_0^t g(c(\tau))\,\mathrm{d}\tau$,

$$\tilde{\rho}(E)(c(t) - x(0)) = \tilde{\rho}(1)(c(t) - x(0)) = \sigma(1)\int_{t_0}^{t} g(c(\tau))\,\mathrm{d}\tau,$$

using $Ec(t) = c(t)$. If we take the derivative over time,

$$\begin{aligned} & \tilde{\rho}(1)\dot{c}(t) &=& \sigma(1)g(c(t)) \\ \Leftrightarrow \quad & \rho'(1)\dot{c}(t) &=& \sigma(1)g(c(t)). \end{aligned}$$

Since we would like to have $c(t) = x(t)$, the solution of (ODE), we absolutely need $\rho'(1) = \sigma(1)$.

Appendix E. Missing Theorems and Proofs

E.1. Global uniqueness

If we assume ODE has at least one solution $x(t)$ (Süli and Mayers, 2003, Theorem 12.1), then we can prove that $x(t)$ is unique if the function g is Lipchitz-continuous.

Proposition E.1 *Let some function $u(t)$ and constant $L \geq 0$. If, for all t, we have*

$$u(t) \leq L\int_{t_0}^{t} u(\tau)\,\mathrm{d}\tau.$$

Then, for all t, $u(t) \leq 0$.

Proof We first show that, for all k,

$$u(t) \leq L\int_{t_0}^{t} \frac{(L|t-\tau|)^k}{k!}u(\tau)\,\mathrm{d}\tau,$$

which is true for $k = 0$. Now, we will show the recursion. Assume the property true for k, and we will prove it for $k+1$. Indeed,

$$u(t) \leq L\int_{t_0}^{t} \frac{(L|t-\tau|)^k}{k!}u(\tau)\,\mathrm{d}\tau \leq L^2\int_{t_0}^{t} \frac{(L|t-\tau|)^k}{k!}\int_{t_0}^{\tau} u(s)\,\mathrm{d}s\,\mathrm{d}\tau.$$

In we swap the integrals, we get

$$u(t) \leq L^2 \int_{t_0}^{t} \int_{s}^{t} \frac{(L|t-\tau|)^k}{k!} \,\mathrm{d}\tau \; u(s) \,\mathrm{d}s = L \int_{t_0}^{t} \frac{(L|t-s|)^{k+1}}{(k+1)!} \; u(s) \,\mathrm{d}s.$$

Since, for all $s \in [t_0, t]$,

$$\frac{(L|t-s|)^{k+1}}{(k+1)!} \leq \frac{(L|t-t_0|)^{k+1}}{(k+1)!},$$

we have proven the recursion for all k. Now, using the fact that

$$\lim_{k\to\infty} \frac{(L|t-t_0|)^{k+1}}{(k+1)!} = 0,$$

we have $u(t) \leq 0$, which proves the proposition.

■

Theorem E.2 (Global uniqueness (Cauchy, Lipschitz)) *Consider the differential equation* (ODE). *If $g(x)$ is Lipschitz-continuous with constant L, then, given $x(0) = x_0$, $x(t)$ is unique.*

Proof Assume there exist two different solutions $x_1(t)$ and $x_2(t)$. In that case,

$$x_1(t) - x_2(t) = \int_0^t g(x_1(\tau)) - g(x_2(\tau)) \,\mathrm{d}\tau.$$

If we consider the norm,

$$\begin{aligned} \|x_1(t) - x_2(t)\| &= \| \int_0^t g(x_1(\tau)) - g(x_2(\tau)) \,\mathrm{d}\tau \| \\ &\leq \int_0^t \|g(x_1(\tau)) - g(x_2(\tau))\| \,\mathrm{d}\tau \\ &\leq L \int_0^t \|x_1(t) - x_2(t)\| \,\mathrm{d}\tau. \end{aligned}$$

Using the Proposition E.1, we deduce that

$$\|x_1(t) - x_2(t)\| \leq 0.$$

Because a norm is always positive, the two curves $x_1(t)$ and $x_2(t)$ should be equal. ■

E.2. Contractivity of ODEs with the one-sided Lipschitz condition

Theorem E.3 *Assume that $x_1(t), x_2(t)$ are solutions of ODE with different initial conditions, where $g(x)$ satisfies the one-sided Lipschitz condition*

$$\langle\, g(x) - g(y), x - y \rangle \leq -\mu \|x - y\|^2.$$

Then, $x_1(t)$ and $x_2(t)$ get closer as t increases, more precisely,

$$\|x_1(t) - x_2(t)\|^2 \leq e^{-2\mu t} \|x_1(0) - x_2(0)\|^2.$$

Proof Let $\mathcal{L}(t) = \|x_1(t) - x_2(t)\|^2$ If we derive $\mathcal{L}(t)$ over time,

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\mathcal{L}(t) &= 2\langle x_1(t) - x_2(t), \dot{x}_1(t) - \dot{x}_2(t)\rangle \\ &= 2\langle x_1(t) - x_2(t), g(x_1(t)) - g(x_2(t))\rangle \\ &\leq -2\mu\mathcal{L}(t).\end{aligned}$$

We thus have $\mathcal{L}(t) \leq e^{-2\mu t}\mathcal{L}(0)$, which is the desired result. ■

E.3. Proof of Proposition 1.1

First of all, we need this classical result of convex analysis.

Proposition E.4 *If f is differentiable and strongly-convex of constant $\mu > 0$, then*

$$f(x) - f(x^*) \leq \frac{1}{2\mu}\|\nabla f(x) - \nabla f(x^*)\|^2, \tag{25}$$

$$\mu\|x - y\|^2 \leq \langle \nabla f(x) - \nabla f(y), x - y\rangle. \tag{26}$$

The proof can be found in Nesterov (2013). We can now prove Proposition 1.1:

Proposition *Let f be a L-smooth and μ-strongly convex function and $x_0 \in \mathbf{dom}(f)$. Writing x^* the minimizer of f, the solution $x(t)$ of* (Gradient Flow) *satisfies*

$$\begin{aligned}f(x(t)) - f(x^*) &\leq (f(x_0) - f(x^*)e^{-2\mu t} \\ \|x(t) - x^*\| &\leq \|x_0 - x^*\|e^{-\mu t}.\end{aligned}$$

Proof Indeed, if we derive the right-hand-side of (1),

$$\frac{\mathrm{d}}{\mathrm{d}t}[f(x(t)) - f(x^*)] = \langle \nabla f(x(t)), \dot{x}(t)\rangle = -\|f'(x(t))\|^2.$$

Using inequality (25),

$$\frac{\mathrm{d}}{\mathrm{d}t}[f(x(t)) - f(x^*)] \leq -2\mu[f(x(t)) - f(x^*)].$$

Solving this differential equation leads to the desired result. We can apply a similar technique for the proof of (2), with the help of inequality (26). ■

E.4. Proof of Proposition 2.4

Proposition *A linear multi step method defined by polynomials (ρ, σ) is consistent if and only if*

$$\rho(1) = 0 \qquad \textit{and} \qquad \rho'(1) = \sigma(1).$$

Proof Assume $t_k = kh$. If we expand $g(x(t_k))$ we have

$$g(x(t_k)) = g(x_0) + O(h).$$

If we do the same thing with $x(t_i)$, we have

$$x(t_k) = x_0 + kh\dot{x}(t_0) + O(h^2) = x_0 + khg(x_0) + O(h^2).$$

If we plug these results in the linear multi-step method,

$$T_k(h) = \frac{1}{h}\rho(1)x_0 + (\rho'(1) - \sigma(1))g(x_0) + O(h).$$

The limit is equal to zero if and only if we satisfy (8). ■

E.5. Proof of Proposition 3.1

Proposition *Given constants $0 < \mu \leq L$, a step size $h > 0$ and a linear two-step method defined by (ρ, σ), under the conditions*

$$\begin{aligned}(\rho_1 + \mu h\sigma_1)^2 &\leq 4(\rho_0 + \mu h\sigma_0),\\ (\rho_1 + Lh\sigma_1)^2 &\leq 4(\rho_0 + Lh\sigma_0),\end{aligned}$$

the roots $r_\pm(\lambda)$ of $\pi_{\lambda h}$, defined in (10), are complex conjugate for any $\lambda \in [\mu, L]$. Moreover, the largest modulus root is equal to

$$\max_{\lambda\in[\mu,L]} |r_\pm(\lambda)|^2 = \max\left\{\rho_0 + \mu h\sigma_0,\ \rho_0 + Lh\sigma_0\right\}.$$

Proof We begin by analyzing the roots $r_\pm$ of the generic polynomial

$$z^2 + bz + c,$$

where b and c are real numbers, corresponding to the coefficients of $\pi_{\lambda h}$, i.e. $b = \rho_1 + \lambda h\sigma_1$ and $c = \rho_0 + \lambda h\sigma_0$. If we want complex roots we need to satisfies

$$b^2 \leq 4c.$$

After replacement, we need to satisfy for any $\lambda \in [\mu, L]$,

$$b^2 - 4c \leq 0 \quad \Leftrightarrow \quad (\rho_1 + \lambda h\sigma_1)^2 - 4(\rho_0 + \lambda h\sigma_0) \leq 0.$$

Since the left side this is a convex function in λ, it is equivalent to check only for the extreme values:

$$\begin{aligned}(\rho_1 + \mu h\sigma_1)^2 &\leq 4(\rho_0 + \mu h\sigma_0),\\ (\rho_1 + Lh\sigma_1)^2 &\leq 4(\rho_0 + Lh\sigma_0).\end{aligned}$$

In the complex conjugate case, the roots have the same modulus,

$$|r_\pm(\lambda)|^2 = |c| = |\rho_0 + \lambda h\sigma_0|.$$

Because the function is convex, the maximum is attained for an extreme value of λ,

$$\max_{\lambda\in[\mu,L]} |r_\pm(\lambda)|^2 = \max\left\{\rho_0 + \mu h\sigma_0,\ \rho_0 + Lh\sigma_0\right\},$$

which is the desired result. ■

E.6. Proof of Proposition 5.1

Proposition *Let f be L-smooth and convex and $x(t)$ be the solution of* (Gradient Flow)*. Then*

$$f(x(t)) - f(x^*) \leq \frac{1}{\frac{t}{\|x_0 - x^*\|^2} + \frac{1}{f(x_0) - f(x^*)}} \leq \frac{\|x_0 - x^*\|^2}{t + (2/L)}. \tag{27}$$

Proof Let $\mathcal{L}(x(t)) = f(x(t)) - f(x^*)$. We notice that $\nabla \mathcal{L}(x(t)) = \nabla f(x(t))$ and $\mathrm{d}\mathcal{L}(x(t))/\,\mathrm{d}t = -\|\nabla f(x(t))\|^2$. Since f is convex,

$$\begin{aligned} \mathcal{L}(x(t)) &\leq \langle \nabla f(x(t)), x(t) - x^* \rangle \\ &\leq \|\nabla f(x(t))\| \|x(t) - x^*\|. \end{aligned}$$

By consequence,

$$-\|\nabla f(x(t))\|^2 \leq -\frac{\mathcal{L}(x(t))^2}{\|x(t) - x^*\|^2} \leq -\frac{\mathcal{L}(x(t))^2}{\|x_0 - x^*\|^2}. \tag{28}$$

The last inequality comes from the fact that $\|x(t) - x^*\|$ decreases over time,

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\|x(t) - x^*\|^2 &= 2\langle \dot{x}(t), x(t) - x^* \rangle, \\ &= -2\langle \nabla f(x(t)), x(t) - x^* \rangle, \\ &\leq 0 \quad \text{since } f \text{ is convex.} \end{aligned}$$

From (28), we deduce the differential inequality

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathcal{L}(x(t)) \leq -\frac{\mathcal{L}(x(t))^2}{\|x_0 - x^*\|^2}.$$

The solution is obtained by integration,

$$\int_0^t \frac{\mathrm{d}\mathcal{L}(x(\tau))/\,\mathrm{d}\tau}{\mathcal{L}(x(\tau))^2}\,\mathrm{d}\tau \leq \int_0^t \frac{-1}{\|x_0 - x^*\|^2}.$$

The general solution is thus

$$\mathcal{L}(x(t)) \leq \frac{1}{\frac{t}{\|x_0 - x^*\|^2} + C},$$

for some constant C. Since the inequality is valid for all time $t \geq 0$, the following condition on C,

$$\mathcal{L}(x(t)) \leq \frac{1}{\frac{t}{\|x_0 - x^*\|^2} + C} \leq \frac{1}{C} \quad \text{for} \quad t \geq 0,$$

is sufficient. Setting $C = \frac{1}{f(x_0) - f(x^*)}$ satisfies the above inequality. If we use the fact that the function is smooth,

$$f(x(t)) - f(x^*) \leq \frac{L}{2}\|x_0 - x^*\|^2,$$

then we get the desired result. ■