Bachelor's Degree Thesis

Discretization of Optimization Flows through State-Triggered Control

Pol Mestres Ramon

Advised by:
Jorge Cortés (UCSD)
Domingo Biel Solé (UPC)

In partial fulfillment of the requirements for the
Bachelor's Degree in Mathematics
Bachelor's Degree in Engineering Physics

May 2020

Abstract

Recent interest in first-order optimization algorithms has lead to the formulation of so-called high-resolution differential equations, continuous surrogates for some of the classical discrete-time optimization algorithms exhibiting acceleration. The continuous setting allows the use of some powerful and well-established tools, like Lyapunov functions. This framework can be used to gain some intuition into the still somewhat mysterious phenomenon of acceleration. In this work we study these high-resolution differential equations and the crucial problem of re-discretizing them while maintaining their rate of convergence. We also review a recent paper that proposes a discretization technique by using ideas borrowed from event-triggered control and suggest some improvements on those algorithms in terms of their rate of convergence.

Keywords: First-Order Optimization, Event-Triggered Control, Nesterov's Accelerated Method, Estimating Sequences, High-Resolution Differential Equations

AMS Code: 37N40

Acknowledgements

First of all, I would like to sincerely thank Prof. Jorge Cortés for giving me the chance to do my Bachelor's Thesis in his lab at UCSD. The weekly meetings and the career advice have been enlightening. I would like to extend my gratitude to Miguel Vaquero, without whom most of this thesis would not have been possible. Thank you for the constructive comments and for giving me the chance to be part of my first scientific publication.
This experience abroad could not have been possible without the personal and financial support of CFIS. I would like to especially thank Prof.Miguel Ángel Barja and Prof.Toni Pascual. I would also like to acknowledge the help of all the people from UPC involved in this work, and in particular my advisor, Domingo Biel.
This thesis represents the culmination of an intellectual and personal endeavour that started in 2015 and that has brought me to meet great people. Without my double-degree colleagues none of this would have been the same. I love you all.
I would also like to thank all the people I have met along the way during these six months in San Diego, specially my roommates (Melcior, Diego, Johannes and Víctor). You have turned these six months into one of the best experiences of my life so far.
Finally, I would like to acknowledge the support of my family during all these years, and for taking care of me during this horrible quarantine.

Pol Mestres Ramon
Llagostera, May 2020

Contents

Chapter 1

Introduction

Optimization has always been at the core of most of the applications of mathematics. In the last decade, Machine Learning and Deep Learning have become the major areas of application of optimization. The types of problems where Machine Learning and Deep Learning are applied are diverse, but all have a common denominator: the requirement to deal with large amounts of data. This demand has direct implications on the types of algorithms that can be implemented. Not only do they have to be *fast*, meaning that they have to reach the optimum with the least number of iterations possible, but also *efficient*, in terms of the amount of storage used.
The above requirements have lead researchers to work in a the framework of *first-order* optimization methods.
This means that at each iteration the algorithm can only use the gradient and the value of the objective function at the point where the algorithm finds itself. No more higher-order information is considered. In particular, this means that common optimization algorithms, like Newton's Method, are out of the scope of this framework.
First-order algorithms allow the storage capabilities to be relatively small, since the gradient of a multivariate function grows linearly with respect to the dimension of the variable to be optimized, whereas the Hessian, for instance, grows quadratically. Throughout this thesis we are going to work in this *first-order* framework.
We will be considering unconstrained minimization problems:

$$\min_{x \in \mathbb{R}^n} f(x)$$

where f is a smooth convex (or strongly-convex) function.

The most common *first-order* method is gradient descent, which leverages the intuition that the direction in which f will decrease the most is that of its gradient. Due to its simplicity, gradient descent (and specially its stochastic version, Stochastic Gradient Descent) is still one of the most widely used methods in the optimization community.

The first improvement on gradient descent dates back to Polyak [13], who presented the *heavy-ball* method, which introduces what is known as a *momentum* term to the gradient step. The term *momentum* comes from an analogy from physics, since the added term is proportional to the *velocity* that a hypothetical particle following that trajectory would have. Polyak's *heavy-ball* is only provably faster than gradient descent *locally* (with quadratic worst-case convergence rate), but it can even fail to converge *globally* for certain functions. When a certain *first-order* optimization algorithm has a better convergence rate than gradient descent we say that it has *acceleration*.

The next major breakthrough in first-order optimization was due to Nesterov [11], who leveraged the momentum ideas from Polyak and developed a method which is faster than gradient descent globally. Moreover, he developed a technique known as *estimating sequences* to prove that his algorithm was *optimal* among first-order methods. Nesterov's estimating sequences are often seen as obscure and relying on an algebraic trick. With the recent interest on first-order optimization methods, researchers have been trying to gain a better, more intuitive understanding of the phenomenon of acceleration.

We will review all of these algorithms and their convergence properties in Chapter 2.

In chapters 4 and 5 we will explore a recent body of work, which uses differential equations that are continuous counterparts of the discrete-time optimization algorithms described above. This translation enables the use of some of the well-known powerful tools of the analysis of nonlinear systems, like Lyapunov functions. The most relevant line of research for us is the one initiated in [17], which introduces a second-order differential equation which is the continuous-time limit of Nesterov's accelerated gradient method, and further expanded in [5], where *high-resolution* differential equations were introduced. These are a more accurate continuous-time limit of the *heavy-ball* and Nesterov methods.

A pivotal question to understand acceleration with which researchers have recently been struggling is that of discretizing these differential equations while maintaining their convergence properties. Numerous discretizations have been tried. In [8] it is shown that high-order Runge Kutta integrators

can be used to discretize Nesterov's continuous model and still retain acceleration. In [4], explicit Euler, implicit Euler and symplectic integrators are applied to the *high-resolution* differential equations and the properties of these discretizations are analyzed. In Chapter 6 we introduce a technique taken from [16] to discretize the *heavy-ball high-resolution* differential equation by using *event-triggered control* (the basic ideas behind it are introduced at Chapter 3). We build on the Lyapunov functions used on previous works to identify triggering conditions that determine the next stepsize as a function of the current iterate. By design, these triggers ensure that the discretization retains the decay rate of the Lyapunov function in the continuous dynamics. In Chapter 7 we build on the approach developed in Chapter 6 to introduce three novel directions via which the convergence-rate of the triggered dynamics can be improved. Some theoretical results and simulations showing this improvement are presented.
Finally, Chapter 8 presents a continuous-time analogue of the *estimating sequences* technique introduced by Nesterov and we try to analyze the phenomenon of *acceleration* from that perspective.
Chapter 6 is the central one in this thesis. The chapters before it can be thought of as a review of the literature necessary to understand it and the chapters after it are build upon the ideas introduced in it.

Chapter 2

Classical Convex Optimization Algorithms

2.1 The gradient method

In the following we study the convergence properties of the most popular convex optimization algorithm: gradient descent. We follow the exposition in [12].
Let us introduce the space of differentiable convex functions with L-Lipschitz continuous gradient, which we denote by $\mathscr{F}_L^{1,1}$. If $f \in \mathscr{F}_L^{1,1}(\mathbb{R}^n)$ then the following inequalities hold for all $x, y \in \mathbb{R}^n$:

$$\langle \nabla f(x) - \nabla f(y), x - y \rangle \geq 0$$
$$\|\nabla f(x) - \nabla f(y)\| \leq L \|x - y\|$$

Let's recall the Gradient (or Gradient Descent) Method to solve the following problem:

$$\min_{x \in \mathbb{R}^n} f(x)$$

Algorithm 0: Gradient Method

Initialization: Initial point ($x_0 \in \mathbb{R}^n$), objective function (f), tolerance (ϵ), stepsizes $\{h_k\}_{k\in\mathbb{N}}$.;
Set: $k = 0$;
while $\|\nabla f(x)\| \geq \epsilon$ **do**
    Compute $f(x_x)$ and $\nabla f(x_k)$;
    Compute next iterate according $x_{k+1} = x_k - h_k \nabla f(x_k)$;
    Set $k = k + 1$
end

Now we focus on the case $h_k = h > 0\ \forall k$. Let x_* be the optimizer. Then:

Theorem 1 ([12])**.** *Let $f \in \mathscr{F}_L^{1,1}(\mathbb{R}^n)$ and $0 < h < \frac{2}{L}$. Then the Gradient Method generates a sequence of points x_k with function values satisfying:*

$$f(x_k) - f(x_*) \leq \frac{2(f(x_0) - f(x_*))\,\|x_0 - x_*\|^2}{2\,\|x_0 - x_*\|^2 + kh(2 - Lh)(f(x_0) - f(x_*))}$$

$\forall k \geq 0$

Proof. Let $r_k = \|x_k - x_*\|$. Then:

$$\begin{aligned} r_{k+1}^2 &= \|x_k - x_* - h\nabla f(x_k)\|^2 \\ &= r_k^2 - 2h\langle \nabla f(x_k), x_k - x_*\rangle + h^2\,\|\nabla f(x_k)\|^2 \\ &\leq r_k^2 - h(\frac{2}{L} - h)\,\|\nabla f(x_k)\|^2 \end{aligned}$$

where we have used property 7 of 41 from Appendix A. Therefore, $r_k \leq r_0$ Now, by using property 3 of 41 from A we have:

$$\begin{aligned} f(x_{k+1}) &\leq f(x_k) + \langle \nabla f(x_k), x_{k+1} - x_k\rangle + \frac{L}{2}\,\|x_{k+1} - x_k\|^2 \\ &= f(x_k) - h(1 - \frac{L}{2}h)\,\|\nabla f(x_k)\|^2 \end{aligned}$$

By convexity:

$$f(x_k) - f(x_*) \leq \langle \nabla f(x_k), x_k - x_*\rangle \langle r_0\,\|\nabla f(x_k)\|$$

After some algebraic manipulation we get:

$$\frac{1}{f(x_{k+1}) - f(x_*)} \geq \frac{1}{f(x_0) - f(x_*)} + \frac{h(1 - \frac{L}{2}h)}{r_0^2}(k+1)$$

which is equivalent to the statement of the theorem. □

Remark 1. *In order to choose the optimal step size, we need to maximize the function $\phi(h) = h(2 - Lh)$ with respect to h. It's easy to check that the maximum is achieved for $h = \frac{1}{L}$. In this case, we get the following convergence rate:*

$$f(x_k) - f(x_*) = \frac{2L(f(x_0) - f(x_*)) \|x_0 - x_*\|^2}{2L\|x_0 - x_*\|^2 + k(f(x_0) - f(x_*)}$$

Let us estimate now the performance of the Gradient Method on the class of strongly convex functions.

We denote by $\mathscr{S}^{1,1}_{\mu,L}$ the class of μ-strongly convex functions with L-Lipschitz-continuous gradients, i.e, functions that satisfy:

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$$
$$f(y) \geq f(x) + \nabla f(x)^T(y - x) + \frac{\mu}{2}\|y - x\|^2$$

Theorem 2 ([12])**.** *If $f \in \mathscr{S}^{1,1}_{\mu,L}$ and $0 < h < \frac{2}{h+L}$ then the Gradient Method generates a sequence x_k such that:*

$$\|x_k - x_*\|^2 \leq (1 - \frac{2h\mu L}{\mu + L})^k \|x_0 - x_*\|^2$$

If $h = \frac{2}{\mu+L}$ (the case for which we reach the highest rate of convergence), then

$$\|x_k - x_*\| \leq (\frac{Q_f - 1}{Q_f + 1})^k \|x_0 - x_*\|$$
$$f(x_k) - f(x_*) \leq \frac{L}{2}(\frac{Q_f - 1}{Q_f + 1})^{2k} \|x_0 - x_*\|^2$$

where $Q_f = \frac{L}{\mu}$ is the so-called conditioning number of f

Proof. Let $r_k = \|x_k - x_*\|$. Then:

$$r^2_{k+1} = \|x_k - x_* - h\nabla f(x_k)\|^2 = r^2_k - 2h\langle \nabla f(x_k), x_k - x_*\rangle + h^2\|\nabla f(x_k)\|^2$$
$$\leq (1 - \frac{2h\mu L}{\mu + L})r^2_k + h(h - \frac{2}{\mu + L})\|\nabla f(x_k)\|^2$$

where we have used the following inequality, the proof of which is presented in A

$$\langle \nabla f(x) - \nabla f(y), x - y\rangle \geq \frac{\mu L}{\mu + L}\|x - y\|^2 + \frac{1}{\mu + L}\|\nabla f(x) - \nabla f(y)\|^2$$

□

2.2 Polyak's Heavy-Ball method

The first improvement to the Gradient Method dates back to Polyak [13], who realized that the convergence can be improved by adding information about previous steps. The *heavy-ball* method incorporates a momentum term to the gradient step, and the iterates become:

$$x_{k+1} = x_k - \alpha \nabla f(x_k) + \beta(x_k - x_{k-1}) \tag{2.1}$$

where $\alpha,\ \beta > 0$ are the stepsize and momentum coefficients, respectively.

Polyak was able to prove, via an eigenvalue argument, that for quadratic functions the rate of convergence improves upon gradient descent. He was also able to prove that the rate of *local* convergence near the objective function's minimum is better than gradient descent. Nevertheless, Polyak was unable to extend the argument *globally*. In fact, there exist examples in the literature where the method even fails to converge. In subsection 2.3 we will take a look at one of those examples.

Next we outline the proof that Polyak's method is faster than gradient descent in the case of quadratic objectives. The proof is essentially the same as the way Polyak presented it in [13] but is extracted from [14]. Assume we aim to minimize $f : \mathbb{R}^n \to \mathbb{R}$ given by

$$f(x) = \frac{1}{2} x^T A x - b^T x + c$$

where A is an $n \times n$ positive definite matrix, b is a vector and c is a constant. We assume that $\mu I \preceq A \preceq LI$

The *heavy-ball* updates can be given in matrix form by:

$$\begin{bmatrix} x_{k+1} - x_* \\ x_k - x_* \end{bmatrix} = \begin{bmatrix} (1+\beta)I - \alpha A & -\beta I \\ I & 0 \end{bmatrix} \begin{bmatrix} x_k - x_* \\ x_{k-1} - x_* \end{bmatrix}$$

Hence,

$$\left\| \begin{bmatrix} x_{k+1} - x_* \\ x_k - x_* \end{bmatrix} \right\| = \left\| T^k \right\| \left\| \begin{bmatrix} x_1 - x_* \\ x_0 - x_* \end{bmatrix} \right\|$$

where

$$T = \begin{bmatrix} (1+\beta)I - \alpha A & -\beta I \\ I & 0 \end{bmatrix}$$

So it suffices to bound the norm of T^k to get a convergence rate. To do so, we use the following result from matrix analysis:

Proposition 3. *Let M be an $n \times n$ matrix. Let $\{\lambda_i(M)\}$ be its eigenvalues. Let $\rho(M) = \max_i |\lambda_i(M)|$. Then there exists a sequence of $\epsilon_k \geq 0$ such that:*

$$\left\|M^k\right\| \leq (\rho(M) + \epsilon_k)^k$$

and $\lim_{k\to\infty} \epsilon_k = 0$.

Proposition 4. *For $\beta \geq \max\{|1 - \sqrt{\alpha\mu}|, |1 - \sqrt{\alpha L}|\}$, $\rho(T) \leq \beta$*

Proof. Let $U\Lambda U^T$ be an eigendecomposition of A. Let Π be the $2n \times 2n$ matrix with entries:

$$\Pi_{i,j} = \begin{cases} 1 & i \text{ odd},\ j = i \\ 1 & i \text{ even},\ j = 2n + i \\ 0 & \text{otherwise} \end{cases}$$

Then,

$$\begin{aligned} &\Pi \begin{bmatrix} U & 0 \\ 0 & U \end{bmatrix}^T \begin{bmatrix} (1+\beta)I - \alpha A & -\beta I \\ I & 0 \end{bmatrix} \begin{bmatrix} U & 0 \\ 0 & U \end{bmatrix} \Pi^T \\ &= \Pi \begin{bmatrix} (1+\beta)I - \alpha\Lambda & -\beta I \\ I & 0 \end{bmatrix} \Pi^T \\ &= \begin{bmatrix} T_1 & 0 & \dots & 0 \\ 0 & T_2 & \dots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & T_n \end{bmatrix} \end{aligned}$$

Where

$$T_i = \begin{bmatrix} 1 + \beta - \alpha\lambda_i & -\beta \\ 1 & 0 \end{bmatrix}$$

Hence, T is similar to the block diagonal matrix with 2×2 diagonal blocks T_i. So to compute the eigenvalues of T it suffices to compute the eigenvalues of all the T_i. The eigenvalues of T_i are the solutions of

$$u^2 - (1 + \beta - \alpha\lambda_i)u + \beta = 0$$

It can be shown that if $\beta \geq (1 - \sqrt{\alpha\lambda_i})^2$, the roots of the characteristic equations are imaginary with magnitude β. Also note that

$$(1 - \sqrt{\alpha\lambda_i})^2 \leq \max\{(1 - \sqrt{\alpha\mu})^2, (1 - \sqrt{\alpha L})^2\}$$

and therefore setting $\beta = \max\{(1 - \sqrt{\alpha\mu})^2, (1 - \sqrt{\alpha L})^2\}$ the proof is complete.

□

In order to find the best possible convergence rate we need to find an α that minimizes $\max\{(1 - \sqrt{\alpha\mu})^2, (1 - \sqrt{\alpha L})^2\}$. This happens when $(1 - \sqrt{\alpha\mu})^2 = (1 - \sqrt{\alpha L})^2$, cf 2.1. We get $\alpha = \frac{4}{(\sqrt{L}+\sqrt{\mu})^2}$ and $\beta = \frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}$.

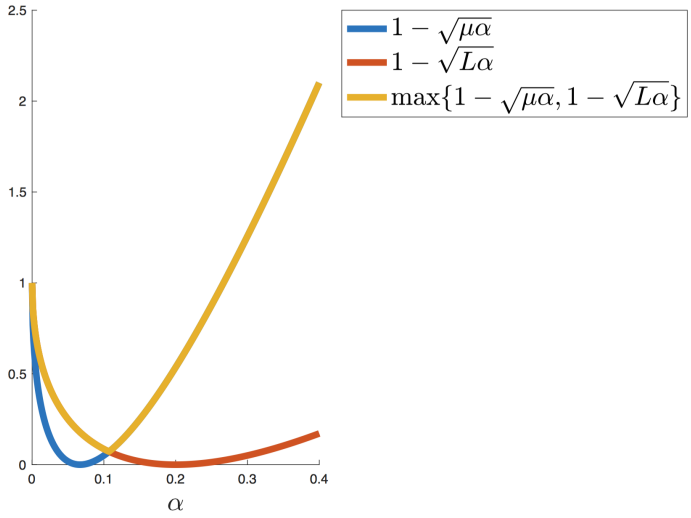

Figure 2.1: $\min_\alpha\{\max\{(1 - \sqrt{\alpha\mu})^2, (1 - \sqrt{\alpha L})^2\}\}$

which leads the following convergence rate:

$$\|x_k - x_*\| \leq (\frac{\sqrt{L} - \sqrt{\mu}}{\sqrt{L} + \sqrt{\mu}} + \epsilon_k)^k D_0$$

where $\lim_{k\to\infty} = 0$ and D_0 is a constant that depends on the initial conditions.

2.2.1 Simulations

In the following figure we can see that after running 7 iterations of both Polyak's *heavy-ball* and Gradient Descent for the objective $f(x_1, x_2) = 5x_1^2 + x_2^2$ the former is much closer to the optimizer (the origin) than the latter:

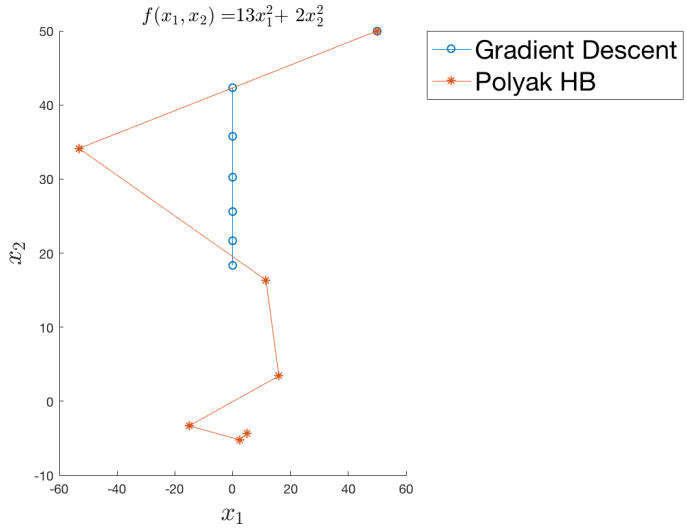

Figure 2.2: Iterates of Gradient Descent and Polyak's *heavy-ball* for the objective $f(x_1, x_2) = 13x_1^2 + 2x_2^2$

2.2.2 Failing Case of Polyak's Momentum

In their 2015 paper, Lessard et al. [9] were able to come up with a convex function and specific hyperparameters of the *heavy-ball* algorithm for which the algorithm fails to converge.
Let f be defined by its gradient by:

$$\nabla f(x) = \begin{cases} 25x & \text{if} x < 1 \\ x + 24 & \text{if} 1 \leq x < 2 \\ 25x - 24 & \text{otherwise} \end{cases}$$

We are not going to get into the technicalities of the proof of why this particular function doesn't converge if we follow Polyak's *heavy-ball* method,

but the basic idea is that the iterates get stuck into a limit cycle. Figure 2.3 shows this behavior.

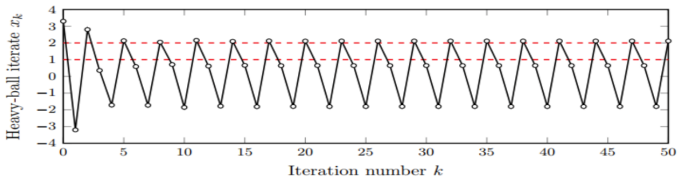

Figure 2.3: Value of f in terms of the number of iteration

2.3 Nesterov's accelerated gradient

In the last section we saw that Polyak's algorithm can fail to converge for some convex functions. Leveraging the idea of momentum introduced by Polyak, Nesterov [11] introduced an algorithm that also presents acceleration and moreover converges for general convex functions:

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

where α and β are hyperparameters of the algorithm.
We can rewrite the iteration in terms of only x_k:

$$x_{k+1} = x_k + \beta(x_k - x_{k-1}) - \alpha \nabla f(x_k + \mu(x_k - x_{k-1})$$

Comparing with Equation 2.1, we see that Polyak's method evaluates the gradient before adding momentum while Nesterov's method evaluates it after adding momentum.

For μ-strongly convex functions with L-Lipschitz gradient, Nesterov considers the algorithm with $\alpha = s$, $\beta = \frac{1-\sqrt{\mu s}}{1+\sqrt{\mu s}}$

$$\begin{aligned} y_{k+1} &= x_k - s \nabla f(x_k) \\ x_{k+1} &= y_{k+1} + \frac{1 - \sqrt{\mu s}}{1 + \sqrt{\mu s}}(y_{k+1} - y_k) \end{aligned}$$

starting from x_0 and $x_1 = x_0 - \frac{2s\nabla f(x_0)}{1+\sqrt{\mu s}}$ and s satisfying $0 \leq s \leq 1/L$. Nesterov proves that this algorithm achieves an accelerated linear convergence rate:

$$f(x_k) - f(x_*) \leq \mathcal{O}\Big(1 - \sqrt{\mu s}\Big)^k\Big)$$

For (weakly) convex functions, Nesterov considers the algorithm with $\alpha = s$ and $\beta = \frac{k}{k+3}$

$$y_{k+1} = x_k - s\nabla f(x_k)$$
$$x_{k+1} = y_{k+1} + \frac{k}{k+3}(y_{k+1} - y_k)$$

with $x_0 = y_0 \in \mathbb{R}^n$ and s satisfying $0 \leq s \leq 1/L$. Nesterov proves the following convergence rate:

$$f(x_k) - f(x_*) \leq \mathcal{O}(\frac{1}{sk^2})$$

Nesterov's proofs of accelerated convergence rates are often regarded as obscure and relying on algebraic tricks, and to the present day there is no clear intuition on why the hyperparameter selection done above leads to *acceleration*. Here we will briefly outline the ideas behind those proofs, which can be found extensively in [12]. The core concept behind those proofs is that of *estimating sequence*, which we define next:

Definition 5. *A pair of sequences $\{\phi_k(x)\}_{k=0}^{\infty}$ and $\{\lambda_k\}_{k=0}^{\infty}$, $\lambda_k \geq 0$ are called estimating sequences of the function f if*

$$\lambda_k \to 0$$

and for any $x \in \mathbb{R}^n$ and all $k \geq 0$ we have

$$\phi_k(x) \leq (1 - \lambda_k)f(x) + \lambda_k\phi_0(x)$$

The next lemma explains the motivation of such definition:

Lemma 6 ([12])**.** *If for some sequence of points $\{x_k\}$ we have:*

$$f(x_k) \leq \phi_k^* \overset{\text{def}}{=} \min_{x\in\mathbb{R}^n} \phi_k(x) \tag{2.2}$$

then $f(x_k) - f(x_*) \leq \lambda_k(\phi_0(x_*) - f(x_*)) \to 0$

Proof.

$$\begin{aligned} f(x_k) \leq \phi_k^* = \min_{x \in \mathbb{R}^n} \phi_k(x) &= \min_{x \in \mathbb{R}^n} \Big((1-\lambda_k) f(x) + \lambda_k \phi_0(x) \Big) \\ &\leq (1-\lambda_k) f(x_*) + \lambda_k \phi_0(x_*) \end{aligned}$$

□

Hence, for any sequence $\{x_k\}$, we can derive its rate of convergence from the sequence $\{\lambda_k\}$. Before we are able to do that, though, we face two problems: how to form the estimating sequences and how to satisfy the inequality 2.2.

Nesterov gives a recursive way to find an estimating sequence:

Lemma 7 ([12])**.** *Assume that:*

- *f is a function belonging to the class $\mathscr{S}_{\mu,L}^{1,1}$,*
- *ϕ_0 is an arbitrary convex function on $\mathbb{R}^n$,*
- *$\{y_k\}_{k=0}^{\infty}$ is an arbitrary sequence of points in $\mathbb{R}^n$*
- *the coefficients $\{\alpha_k\}_{k=0}^{\infty}$ satisfy conditions $\alpha_k \in (0,1)$ and $\sum_{k=0}^{\infty} \alpha_k = \infty$*
- *we choose $\lambda_0 = 1$*

Then the pair of sequences $\{\phi_k\}_{k=0}^{\infty}$ and $\{\lambda_k\}_{k=0}^{\infty}$, defined recursively by the relations

$$\begin{aligned} \lambda_{k+1} &= (1-\alpha_k)\lambda_k \\ \phi_{k+1}(x) &= (1-\alpha_k)\phi_k(x) + \alpha_k \Big(f(y_k) + \langle \nabla f(y_k), x - y_k \rangle + \frac{\mu}{2} \|x - y_k\|^2 \Big) \end{aligned} \tag{2.3}$$

are estimating sequences

The above lemma allows us to update the estimating sequences in terms of an arbitrary sequence of points and an arbitrary sequence of coefficients (satisfying the conditions outlined). Note that we are also free to choose the initial function $\phi_0(x)$. The following lemma establishes that if we choose $\phi_0(x)$ of a particular form, then the $\phi_k(x)$ have a canonical form for all k

Lemma 8 ([12]). *Let $\phi_0(x) = \phi_0^* + \frac{\gamma_0}{2}\|x - v_0\|^2$. Then the process 2.3 preserves the canonical form of functions $\{\phi_k(x)\}$:*

$$\phi_k(x) \equiv \phi_k^* + \frac{\gamma_k}{2}\|x - v_k\|^2$$

where the sequences $\{\gamma_k\}$, $\{v_k\}$ and $\{\phi_k^*\}$ are defined as follows:

$$\begin{aligned}
\gamma_{k+1} &= (1-\alpha_k)\gamma_k + \alpha_k\mu \\
v_{k+1} &= \frac{1}{\gamma_{k+1}}\Big((1-\alpha_k)\gamma_k v_k + \alpha_k\mu y_k - \alpha_k\nabla f(y_k)\Big) \\
\phi_{k+1}^* &= (1-\alpha_k)\phi_k^* + \alpha_k f(y_k) - \frac{\alpha_k^2}{2\gamma_{k+1}}\|\nabla f(y_k)\|^2 \\
&\quad + \frac{\alpha_k(1-\alpha_k)\gamma_k}{\gamma_{k+1}}\Big(\frac{\mu}{2}\|y_k - v_k\|^2 + \langle\nabla f(y_k), v_k - y_k\rangle\Big)
\end{aligned}$$

By having the ϕ_k defined that way, we are closer to getting an algorithmic scheme. Suppose, for induction's sake, that we already have x_k satisfying:

$$\phi_k^* \geq f(x_k)$$

Our goal is to define an x_{k+1} such that $\phi_{k+1}^* \geq f(x_{k+1})$ Then, by Lemma 8,

$$\begin{aligned}
\phi_{k+1}^* \geq (1-\alpha_k)f(x_k) + \alpha_k f(y_k) - \frac{\alpha_k^2}{2\gamma_{k+1}}\|\nabla f(y_k)\|^2 \\
\frac{\alpha_k(1-\alpha_k)\gamma_k}{\gamma_{k+1}}\langle\nabla f(y_k), v_k - y_k\rangle
\end{aligned}$$

Since f is convex, $f(x_k) \geq f(y_k) + \langle\nabla f(y_k), x_k - y_k\rangle$, we get the following estimate:

$$\begin{aligned}
\phi_{k+1}^* &\geq f(y_k) - \frac{\alpha_k^2}{2\gamma_{k+1}}\|\nabla f(y_k)\|^2 \\
&\quad + (1-\alpha_k)\langle\nabla f(y_k), \frac{\alpha_k\gamma_k}{\gamma_{k+1}}(v_k - y_k) + x_k - y_k\rangle
\end{aligned}$$

Recall that we want to ensure $\phi_{k+1}^* \geq f(x_{k+1})$. Recall also that we can ensure the inequality

$$f(y_k) - \frac{1}{2L}\|\nabla f(y_k)\|^2 \geq f(x_{k+1})$$

by taking the gradient step

$$x_{k+1} = y_k - \frac{1}{L}\nabla f(y_k)$$

Define α_k as the positive root of the quadratic

$$L\alpha_k^2 = (1-\alpha_k)\gamma_k + \alpha_k\mu = \gamma_{k+1}$$

Then $\frac{\alpha_k^2}{2\gamma_{k+1}} = \frac{1}{2L}$ and we can replace the previous inequality by the following one:

$$\phi_{k+1}^* \geq f(x_{k+1}) + (1-\alpha_k)\langle \nabla f(y_k), \frac{\alpha_k\gamma_k}{\gamma_{k+1}}(v_k - y_k) + x_k - y_k\rangle$$

Now let's choose y_k so that the extra term on the right hand side vanishes:

$$\frac{\alpha_k\gamma_k}{\gamma_{k+1}}(v_k - y_k) + x_k - y_k = 0$$

which leads to $y_k = \frac{\alpha_k\gamma_k v_k + \gamma_{k+1}x_k}{\gamma_k + \alpha_k\mu}$

The discussion we have just outlined is formalized in the following algorithm, which is often called the *Fast Gradient Method*

Algorithm 1: Estimating Sequences Algorithm

Initialization: Initial point ($x_0 \in \mathbb{R}^n$), some $\gamma_0 > 0$ and $v_0 = x_0$;
Set: $k = 0$;
while $\|\nabla f(x_k)\| \geq \epsilon$ **do**
  Compute $\alpha_k \in (0,1)$ from the equation $L\alpha_k^2 = (1-\alpha_k)\gamma_k + \alpha_k\mu$;
  Set $\gamma_{k+1} = (1-\alpha_k)\gamma_k + \alpha_k\mu$;
  Choose $y_k = \frac{\alpha_k\gamma_k v_k + \gamma_{k+1}x_k}{\gamma_k + \alpha_k\mu}$. Compute $f(y_k)$ and $\nabla f(y_k)$;
  Find x_{k+1} such that $f(x_{k+1}) \leq f(y_k) - \frac{1}{2L}\|\nabla f(y_k)\|^2$ (for example by setting $x_{k+1} = y_k - \frac{1}{L}\nabla f(y_k)$);
  Set $v_{k+1} = \frac{1}{\gamma_{k+1}}\Big((1-\alpha_k)\gamma_k v_k + \alpha_k\mu y_k - \alpha_k\nabla f(y_k)\Big)$;
  Set $k = k+1$
end

From the discussion we did and 2.2 the following theorem follows:

Theorem 9. *Algorithm 1 generates a sequence of points $\{x_k\}$ such that*

$$f(x_k) - f(x_*) \leq \lambda_k\Big(f(x_0) - f(x_*) + \frac{\gamma_0}{2}\|x_0 - x_*\|^2\Big)$$

where $\lambda_0 = 1$ and $\lambda_k = \prod_{i=0}^{k-1}(1-\alpha_i)$

Thus, in order to estimate the rate of convergence of algorithm 1 we need to understand how quickly the sequence $\{\lambda_k\}$ approaches zero. The following lemma characterizes this:

Lemma 10. *If in the algorithm 1 we choose $\gamma_0 \in (\mu, 3L + \mu]$, then for all $k \geq 0$ we have*

$$\lambda_k \leq \frac{4\mu}{(\gamma_0 - \mu)(\Big(exp(\frac{k+1}{2} q_f^{1/2}) - exp(-\frac{k+1}{2} q_f^{1/2})\Big))^2} \leq \frac{4L}{(\gamma_0 - \mu)(k+1)^2}$$

where $q_f = \frac{\mu}{L}$.

In particular, for $\gamma_0 = \mu$, $\lambda_k = (1 - \sqrt{q_f})^k$, $k \geq 0$

Remark 2. *It should be noted that Nesterov actually proves that the rates given in Lemma 10 are optimal for first order methods, i.e, methods that at each iteration only have access to the value of the function and the gradient of the function. We are not going to get into the details of such a proof, but all the details can be found at [12]*

The only question that still remains to be solved by this point is how does Nesterov's algorithm as presented in the beginning of this section relate to algorithm 1. In the following we delve into this issue.

Consider a variant of scheme 1 which uses a constant gradient step for finding the point x_{k+1}, so that $x_{k+1} = x_k - \frac{1}{L}\nabla f(x_k)$. Under this update, it can be shown that algorithm 1 is equivalent to the following one, in which sequences v_k and γ_k have been eliminated.

If we choose $\alpha_0 = \sqrt{q_f}$ (this corresponds to $\gamma_0 = \mu$). Then, $\forall k \geq 0$

$$\alpha_k = \sqrt{q_f}$$
$$\beta_k = \frac{1 - \sqrt{q_f}}{1 + \sqrt{q_f}}$$

which yields Nesterov's algorithm for strongly convex functions as we presented it in the beginning of the section.

Algorithm 2: Estimating Sequences Algorithm Constant Stepsize

Initialization: Initial point ($x_0 \in \mathbb{R}^n$), some $\alpha_0 \in (0,1)$ and $y_0 = x_0$;
Set: $k = 0$;
while $\|\nabla f(x_k)\| \geq \epsilon$ **do**
  Compute $f(y_k)$ and $\nabla f(y_k)$.;
  Set $x_{k+1} = y_k - \frac{1}{L}\nabla f(y_k)$;
  Compute $\alpha_{k+1} \in (0,1)$ from the equation

$$\alpha_{k+1}^2 = (1 - \alpha_{k+1})\alpha_k^2 + q_f \alpha_{k+1}$$

  Set $\beta_k = \frac{\alpha_k(1-\alpha_k)}{\alpha_k^2+\alpha_{k+1}}$;
  Set $y_{k+1} = x_{k+1} + \beta_k(x_{k+1} - x_k)$;
  Set $k = k + 1$
end

2.3.1 Simulations

In the following figure we can see that after running 7 iterations of both Nesterov's Accelerated Gradient method and Gradient Descent for the objective $f(x_1, x_2) = 5x_1^2 + x_2^2$ the former is much closer to the optimizer (the origin) than the latter:

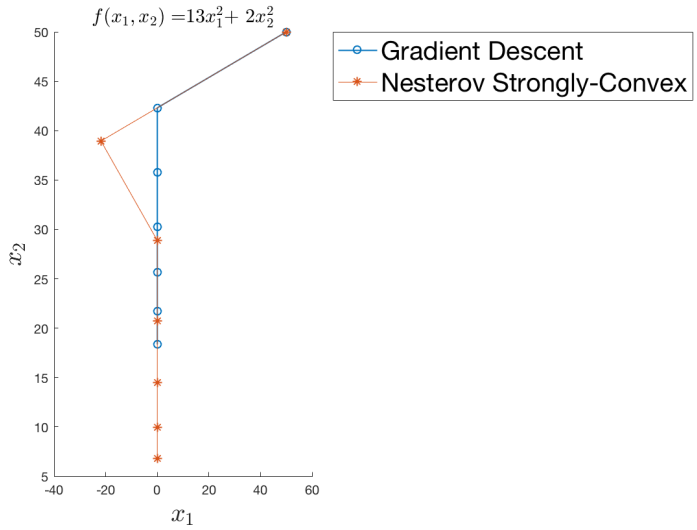

Figure 2.4: Iterates of Gradient Descent and Nesterov's Accelerated Gradient for the objective $f(x_1, x_2) = 13x_1^2 + 2x_2^2$

Chapter 3

Event-Triggered Control

In this chapter we introduce the basics of event-triggered control, which will be applied to optimization algorithms.
The basic idea of event-triggered control is to use aperiodic sampling/control instead of periodic or continuous sampling/control. Instead of continuously or periodically monitoring the state of a control system, the basic assumption behind event-triggered control is that maybe it is more efficient to only do so sporadically, when a certain *triggering* condition is violated (we will define this notion formally later on). The fundamental challenge is to determine precisely when control signals should be updated to improve efficiency while still guaranteeing a desired quality of service (which in practice means satisfying a certain Lyapunov decay). The following exposition is based in [6], which is a tutorial on Event-Triggered Control applied to the problem of multi-agent consensus. Although we will use it for a completely different problem, the basic concepts still apply.

3.1 What is Event-Triggered Control?

Given a control system in $\mathbb{R}^n$ of the form

$$\dot{x} = F(x, u)$$

with an unforced equilibrium at x_*, (i.e., $F(x_*, 0) = 0$), assume we have:

- a continuous-time controller $k : \mathbb{R}^n \to \mathbb{R}^m$

- a certificate of the correctness of the continous-time controller in the form of a Lyapunov function $V : \mathbb{R}^n \to \mathbb{R}$

The previous two points are equivalent to saying that the system $\dot{x} = F(x, k(x))$ makes x_* asymptotically stable, and this fact can be guaranteed by using V as a Lyapunov function.
The idea of event-triggered control is not to continuously update the controller $k(x)$ (as this is infeasible in real life) but rather use a sampled version $\hat{x}$ of the state and update it as $k(\hat{x})$. The question is if we can guarantee (through V) that the sampled controller still stabilizes the system. We also need to find a systematic way to determine the times when the controller must be updated.
By using the sampled controller the closed-loop system is given by:

$$\dot{x} = F(x, k(\hat{x}))$$

More specifically, letting $\{t_l\}_{l \in \mathbb{N}}$ be the sequence of event times at which the control input is updated,

$$u(t) = k(\hat{x}(t))$$

where

$$\hat{x}(t) = x(t_l) \text{ for } t \in [t_l, t_{l+1})$$

Also note that $\dot{V} = \nabla V(x) F(x, k(\hat{x}))$. Moreover, if F is uniformly (in x) Lipschitz in its second argument, k is Lipschitz and ∇V is bounded, the following inequality holds:

$$\dot{V} \leq \nabla V(x) F(x, k(x)) + G(x) \|e\| \tag{3.1}$$

for some function G taking nonnegative values, and where $e = \hat{x} - x$ is the error between the sampled and the actual state.

Since V is a Lyapunov function for the system $\dot{x} = F(x, k(x))$, the first term on the inequality is negative. To ensure that $\dot{V} \leq 0$, we can prescribe a *triggering* condition that resamples the state whenever the first and second terms of the inequality are equal.
In general, a *triggering condition* is encoded via a triggering function f, which

evaluates whether a given state x and error e combination should trigger an event or not. We define the triggering condition as:

$$f(e, w) = g(e) - h(w) = 0 \tag{3.2}$$

where $g : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ is a nonnegative function of the error with $g(0) = 0$ and $h \in \mathbb{R}_{\geq 0}$ is a threshold function that may depend on the state x, the sampled state $\hat{x}$ and time t. When the triggering condition is satisfied, the state is resampled, and the error e is reset to zero. The event times are thus defined as:

$$t_{l+1} = \min\{t^{'} \geq t_l \mid f(e(t^{'}, w(t^{'})) = 0\} \tag{3.3}$$

It is easy to see from equation 3.1 that the functions g and h are given by:

$$g(e) = \|e\|$$
$$h(x) = \frac{|\nabla V(x) F(x, k(x))|}{|G(x)|}$$

By applying the aforementioned triggering condition we can ensure:

$$\dot{V} \leq \nabla V(x) F(x, k(x)) + G(x) \|e\| \leq 0$$

at all times. This implies that x asymptotically approaches x_* as long as the sequence of event times tends to infinity. We formalize this last condition in the following section:

3.2 Zeno behavior

From the discussion we have made in the previous section we know that by following the triggering condition we can ensure that the Lyapunov function is decreasing for all times where the triggering condition is satisfied. It might happen, however, that at some point the times for which the triggering condition is true become arbitrarily close together and the trajectory of the system gets stuck at some point. This is of course an undesirable property in real-life applications, since that would means that the system doesn't reach

the desired equilibrium. The following definition describes the absence of this behavior:

Definition 11. *(Zeno behavior) Given the closed-loop dynamics* $\dot{x} = F(x, k(\hat{x}))$ *driven by 3.3, a solution with initial condition* $x(0) = x_0$ *exhibits Zeno behavior if* $\exists\ T > 0$ *such that* $t_l \leq T\ \forall l \in \mathbb{Z}_{\geq 0}$

In other words, if the event-triggered controller defined by the triggering condition requires an infinite number of events happening in a finite time interval, then the solution exhibits Zeno behavior.
Note also that for a system not to exhibit Zeno behavior, we require solutions for all initial conditions not to have Zeno behavior.
Being able to rule out Zeno behavior is fundamental when it comes to validating the correctness of a given event-triggered controller, because for real world applications it is impossible to update a controller infinitely many times in a bounded period of time.
In practical applications, it might sometimes be hard to prove that a system doesn't exhibit Zeno behavior by proving that the event times aren't uniformly upper bounded. Instead, it might be useful to prove that another quantity, called the *inter-event time*, is positively lower bounded. I.e, proving that $\exists\ \tau^{min}$ such that:

$$t_{l+1} - t_l \geq \tau^{min} > 0$$

$\forall\ l \in \mathbb{N}$. It is clear that this condition is stronger than the lack of Zeno behavior, since

$$\begin{aligned} t_{l+1} - t_l \geq \tau^{min} > 0 \Rightarrow t_l > l\tau^{min} + \tau_0 \\ \Rightarrow \lim_{l\to\infty} t_l = \infty \end{aligned}$$

But it isn't equivalent to it, as the following counterexample shows:

Example 1. *Consider* $t_l = \sum_{k=0}^{l} \frac{1}{k}$. *Since the harmonic sum diverges,* $\lim_{l\to\infty} t_l = \infty$, *but on the other hand,* $t_{l+1} - t_l = \frac{1}{l+1}$ *isn't positively lower bounded.*

This sufficient condition for Zeno behavior will be extensively used in chapters 6 and 7.
Note that even though an Event-Triggered scheme doesn't require to update

the controller continuously, in order to know when to update it, we have to continuously monitor the triggering condition in order to check when it stops being true. Since such a continuous surveillance is again impossible in the real world, in actual applications this condition is monitored periodically.
If defines an implicit equation, an alternative is to solve it numerically by making use of the extensive arsenal of numerical methods used to mind the zeros of nonlinear equations. In the particular case where we can explicitly solve this equation, the controller is known as *self-triggered*. We will find examples of each type throughout the thesis.

Chapter 4

Differential Equations for Optimization Algorithms

In this chapter we will review some of the recent attempts to find an intuitive understanding of *acceleration* of optimization algorithms from the continuous-time perspective.
It should be noted that there is a long history relating ordinary differential equations (ODEs) to optimization. This connection is often obtained by taking increasingly small step sizes so that the points generated by the algorithm converge to a curve modeled by an ODE. By using well-established techniques from ODEs, like Lyapunov theory, many interesting results have been obtained. [7] is a book completely devoted on the topic.

4.1 A Simple Example: Linear Convergence of Gradient Descent via Lyapunov Functions

In this section we show, via a simple example, how the continuous-time limit of an optimization algorithm can be useful in deriving some of its properties. We will consider the most simple out of all first-order optimization algorithms: gradient descent:

$$x_{k+1} = x_k - s\nabla f(x_k)$$

where s is the so-called *step size*. By making s arbitrarily small we end up with the following differential equation:

$$\dot{x} + \nabla f(x) = 0$$

which is often known as the *gradient flow* of f.
In Chapter 2 we prove how this optimization scheme has a convergence rate for convex functions of $\mathcal{O}(1/k)$. Here we will prove, via a Lyapunov function approach, that the solutions of the *gradient flow* converge to the optimum of f at a rate of $\mathcal{O}(1/t)$.

Consider the following functional:

$$V(t) = t(f(x(t)) - f(x^*)) + \frac{1}{2}\left\|x(t) - x_*\right\|^2$$

Let's calculate the derivative of the functional along the solutions of $\dot{x} + \nabla f(x) = 0$

$$\dot{V} = f(x(t)) - f(x_*) - t\left\|\nabla f(x(t))\right\|^2 + \langle x(t) - x_*, -\nabla f(x(t))\rangle$$

Recall that if f is convex, the following inequality is true:

$$f(x_*) \geq f(x(t)) + \langle \nabla f(x(t)), x - x(t)\rangle$$

Therefore, $\dot{V} \leq 0$ along the solutions of the *gradient flow*. This implies:

$$V(t) \leq V(0) = \frac{1}{2}\left\|x(0) - x_*\right\|^2 \tag{4.1}$$

$$\Rightarrow t(f(x(t)) - f(x_*)) + \frac{1}{2}\left\|x(t) - x_*\right\|^2 \leq \frac{1}{2}\left\|x(0) - x_*\right\|^2 \tag{4.2}$$

$$\Rightarrow t(f(x(t)) - f(x_*)) \leq \frac{1}{2}\left\|x(0) - x_*\right\|^2 \tag{4.3}$$

Which leads to the desired convergence rate:

$$f(x(t)) - f(x_*) \leq \frac{\left\|x(0) - x_*\right\|^2}{2t}$$

It is also possible to discretize the functional V and adapt the proof in discrete time to find an alternative proof of the $\mathcal{O}(1/k)$ convergence rate of the Gradient Descent Algorithm.

4.2 A Differential Equation for Modeling Nesterov's Accelerated Gradient Method

Su et al.[17] applied the same ideas to find a differential equation that is the continuous-time limit of Nesterov's Accelerated Gradient scheme. As a byproduct of the derived ODE, a family of schemes with similar convergence rates was found, and some qualitative understanding on the trajectories of Nesterov's scheme was gained.

4.2.1 Deriving the ODE

Recall that Nesterov's scheme for convex functions relies on two sequences, whose updates are given by:

$$x_k = y_{k-1} - s\nabla f(y_{k-1})$$
$$y_k = x_k + \frac{k-1}{k+2}(x_k - x_{k-1})$$

Combining the two equations:

$$\frac{x_{k+1} - x_k}{\sqrt{s}} = \frac{k-1}{k+2}\frac{x_k - x_{k-1}}{\sqrt{s}} - \sqrt{s}\nabla f(y_k) \tag{4.4}$$

Introduce the *ansatz* $x_k \approx X(k\sqrt{s})$ for some smooth curve $X(t)$ defined for $t \geq 0$. Then, as the step size s tends to zero, Taylor expansion gives:

$$\frac{x_{k+1} - x_k}{\sqrt{s}} = \dot{X}(k\sqrt{s}) + \frac{1}{2}\ddot{X}(k\sqrt{s})\sqrt{s} + o(\sqrt{s})$$
$$\frac{x_k - x_{k-1}}{\sqrt{s}} = \dot{X}(k\sqrt{s}) - \frac{1}{2}\ddot{X}(k\sqrt{s})\sqrt{s} + o(\sqrt{s})$$

Note also that

$$y_k = x_k + \frac{k-1}{k+2}(x_k - x_{k-1}) \tag{4.5}$$

$$X(k\sqrt{s}) + \frac{k-1}{k+2}(X(k\sqrt{s}) - X((k-1)\sqrt{s})) = X(k\sqrt{s}) + o(\sqrt{s}) \tag{4.6}$$

and hence, $\sqrt{s}\nabla f(y_k) = \sqrt{s}\nabla f(X(t)) + o(\sqrt{s})$ and 4.4 can be written as:

$$\dot{X}(t)+\frac{1}{2}\ddot{X}(t)\sqrt{s}+o(\sqrt{s}) = (1-\frac{3\sqrt{s}}{t})(\dot{X}(t)-\frac{1}{2}\cdot\cdot\cdot X(t)\sqrt{s}+o(\sqrt{s}))-\sqrt{s}\nabla f(X(t))+o(\sqrt{s}) \tag{4.7}$$

By comparing the cofficients of $\sqrt{s}$ in 4.7 we obtain:

$$\ddot{X} + \frac{3}{t}\dot{X} + \nabla f(X) = 0 \tag{4.8}$$

with initial conditions which can be shown to be $X(0) = x_0$ and $\dot{X}(0) = 0$

4.2.2 Initial Asymptotic

If Nesterov's scheme is implemented, one observes that it seems to move slowly in the beginning, and only after a certain amount of time has passed, the algorithm presents acceleration. We can get some insights on why this phenomenon happens by analyzing the ODE 4.8.
Assume X is smooth enough so that $\lim_{t\to 0}\ddot{X}$ exists. By the mean value theorem there exists some $\xi \in (0,t)$ that satisfies $\frac{\dot{X}(t)-\dot{X}(0)}{t} = \ddot{X}(\xi)$. From the ODE:

$$\ddot{X}(t) + 3\ddot{X}(\xi) + \nabla f(X(t)) = 0$$

Taking the limit $t \to 0$ we get $\ddot{X}(0) = -\frac{\nabla f(x_0)}{4}$ and for small t the solution takes the following form:

$$X(t) = -\frac{\nabla f(x_0)t^2}{8} + x_0 + o(t^2)$$

which is an asymptotic expansion consistent with the empirical observation that Nesterov's scheme moves slowly for the first iterations.

4.2.3 Analogous Convergence Rate

Let's start this subsection by introducing a uniqueness result for 4.8:

Theorem 12 ([17]). *For any $f \in \mathscr{F}_\infty := \cup_{L>0}\mathscr{F}_L$ and any $x_0 \in \mathbb{R}^n$, the ODE 4.8 with initial conditions $X(0) = x_0$, $\dot{X}(0) = 0$ has a unique global solution $X \in C^2((0,\infty);\mathbb{R}^n) \cap C^1([0,\infty);\mathbb{R}^n)$*

Recall that the original result from Nesterov (1983) states that for any function f with Lipschitz-continuous gradient, the iterates of Nesterov's Accelerated Gradient (for convex functions) with step size $s \leq 1/L$ satisfies:

$$f(x_k) - f(x_*) \leq \frac{2\|x_0 - x_*\|^2}{s(k+1)^2}$$

Just like what we did previously for Gradient Descent, the next result indicates that the trajectory of 4.8 converges to the minimizer to the same rate in continuous time, i.e, $f(X(t)) - f(x_*) = \mathcal{O}(\frac{1}{t^2})$

Theorem 13 ([17])**.** *For any $f \in \mathscr{F}_\infty$, let $X(t)$ be the unique global solution to 4.8 with initial conditions $X(0) = x_0$, $\dot{X}(0) = 0$. Then, for any $t > 0$,*

$$f(X(t)) - f(x_*) \leq \frac{2\|x_0 - x_*\|^2}{t^2}$$

Proof. Consider the functional defined by

$$V(t) = t^2(f(X(t)) - f(x_*)) + 2\left\|X + t\dot{X}/2 - x_*\right\|^2$$

Its time derivative is given by:

$$\dot{V} = 2t(f(X) - f(x_*)) + t^2\langle \nabla f, \dot{X}\rangle + 4\Big\langle X + \frac{t}{2}\dot{X} - x_*, \frac{3}{2}\dot{X} + \frac{t}{2}\ddot{X}\Big\rangle$$

Substituting $3\dot{X}/2 + t\ddot{X}/2$ with $-t\nabla f(X)/2$, the time derivative yields:

$$\dot{V} = 2t(f(X) - f(x_*)) - 2t\langle X - x_*, \nabla f(X)\rangle \leq 0$$

where the last inequality follows from the convexity of f. Thus,

$$f(X(t)) - f(x_*) \leq \frac{V(t)}{t^2} \leq \frac{V(0)}{t^2} = \frac{2\|x_0 - x_*\|^2}{t^2}$$

□

4.2.4 A Phase Transition

One of the things that might seem strange from 4.8 is the constant 3 appearing in the coefficient of $\dot{X}$. Su et al.[17] prove that this constant can be replaced by any larger number and the $\mathcal{O}(1/k^2)$ convergence rate is maintained. I.e, the differential equation

$$\ddot{X} + \frac{r}{t}\dot{X} + \nabla f(X) = 0$$

with initial conditions $X(0) = x_0$, $\dot{X}(0) = 0$ has a *phase transition* for $r = 3$, meaning that for $r \geq 3$ a convergence rate of $\mathcal{O}(\frac{1}{k^2}$ and for $r < 3$ the convergence is $\mathcal{O}(\frac{1}{k})$

For the discrete algorithm, this translates to saying that the scheme

$$\begin{aligned} x_k &= y_{k-1} - s\nabla f(y_{k-1}) \\ y_k &= x_k + \frac{k-1}{k+r-1}(x_k - x_{k-1}) \end{aligned}$$

has quadratic convergence only for $r \geq 3$.

4.3 Differential Equations for Nesterov's Accelerated Gradient and Polyak's Heavy Ball for strongly convex functions

One of the points that remains obscure from the exposition on 2 is why Nesterov's Accelerated Gradient achieves accelerated global convergence whereas for Polyak's *heavy-ball* we can only guarantee this result locally.
If we write Nesterov's algorithm for strongly convex function in single-variable form:

$$x_{k+1} = x_k + \frac{1-\sqrt{\mu s}}{1+\sqrt{\mu s}}(x_k - x_{k-1}) - s\nabla f(x_k) - \frac{1-\sqrt{\mu s}}{1+\sqrt{\mu s}}s(\nabla f(x_k) - \nabla f(x_{k-1}))$$

starting from x_0 and $x_1 = x_0 - \frac{2s\nabla f(x_0)}{1+\sqrt{\mu s}}$, we see that the *heavy-ball* method and Nesterov's Accelerated Gradient are identical except for the last term,

$$-\frac{1-\sqrt{\mu s}}{1+\sqrt{\mu s}}s(\nabla f(x_k)-\nabla f(x_{k-1}))$$

the *gradient correction*.

Even though the estimating sequence technique used by Nesterov delivers a proof of acceleration, it does not explain why the absence of the *gradient correction* prevents the heavy-ball method from achieving acceleration for strongly-convex functions.

We would have hope that the continuous setting would shed some light into this issue. Unfortunately, if one follows an analogous derivation to the one we did in 4.2.1, with *heavy-ball* for Nesterov's Accelerated Gradient for Strongly-Convex functions, one finds that the resulting ODE is the same in both cases:

$$\ddot{X}(t)+2\sqrt{\mu}\dot{X}(t)+\nabla f(X(t))=0$$

And therefore, this ODE doesn't provide any insight into why the two algorithms behave differently.

In the work [5], with the goal of finding a different ODE for each scheme, the concept of *high-resolution differential equations* is introduced, which will be the main topic of the next chapter.

Chapter 5

High Resolution Differential Equations

As we pointed out in the previous chapter, a simple continuous-time limit of Nesterov's Accelerated Gradient for strongly-convex functions yields the same ODE as the continuous time limit of Polyak's *heavy-ball* for strongly-convex functions.
However, just as there is not a single preferred way to discretize a differential equation, there is not a preferred way to take a continuous-time limit of a difference equation. Inspired by a common technique used in fluid dynamics, in which physical phenomena are studied at different scales via the inclusion of various orders of perturbations, the authors in [5] propose to incorporate $\mathcal{O}(\sqrt{s})$ terms into the limiting process for obtaining the ODE. This results in what they call *high-resolution* differential equations, which are able to differentiate between the Nesterov Accelerated Gradient (NAG) methods and the *heavy-ball* method. In section 5.1 we show how to derive these *high-resolution* differential equations. Next, in section 5.2 we provide convergence rates of the solutions of these *high-resolution* differential equations. Finally, in section 5.3 we try to translate these convergence rates to the actual discrete-time algorithms we described in Chapter 2. Along the line, we will find some interesting insights on the difference between these two methods.

5.1 Deriving High-Resolution ODEs

We will start by deriving the high-resolution differential equation for Nesterov's Accelerated Gradient for Strongly-Convex functions.
Just like in the *low-resolution* derivation made in [17], our focus is on the single-variable form of the scheme:

$$x_{k+1} = x_k + \frac{1-\sqrt{\mu s}}{1+\sqrt{\mu s}}(x_k - x_{k-1}) - s\nabla f(x_k) - \frac{1-\sqrt{\mu s}}{1+\sqrt{\mu s}}s(\nabla f(x_k) - \nabla f(x_{k-1})) \tag{5.1}$$

Let $t_k = k\sqrt{s}$ and assume there exists a sufficiently smooth curve $X(t)$ such that $x_k = X(t_k)$. Performing a Taylor expansion in powers of $\sqrt{s}$, we get:

$$x_{k+1} = X(t_{k+1}) = X(t_k) + \dot{X}(t_k)\sqrt{s} + \frac{1}{2}\ddot{X}(t_k)(\sqrt{s})^2 + \frac{1}{6}\dddot{X}(t_k)(\sqrt{s})^3 + \mathcal{O}((\sqrt{s})^4)$$
$$x_{k-1} = X(t_{k-1}) = X(t_k) - \dot{X}(t_k)\sqrt{s} + \frac{1}{2}\ddot{X}(t_k)(\sqrt{s})^2 - \frac{1}{6}\dddot{X}(t_k)(\sqrt{s})^3 + \mathcal{O}((\sqrt{s})^4)$$

A Taylor expansion for the *gradient correction* term is the following:

$$\nabla f(x_k) - \nabla f(x_{k-1}) = \nabla^2 f(X(t_k))\dot{X}(t_k)\sqrt{s} + \mathcal{O}(\sqrt{s})^2$$

Substituting into 5.1 and ignoring $\mathcal{O}(s)$ terms but retaining $\mathcal{O}(\sqrt{s})$ we obtain the following *high-resolution* differential equation for Nesterov's Accelerated Gradient for strongly-convex functions:

$$\ddot{X} + 2\sqrt{\mu}\dot{X} + \sqrt{s}\nabla^2 f(X)\dot{X} + (1+\sqrt{\mu s})\nabla f(X) = 0 \tag{5.2}$$

with initial conditions which can be shown to be

$$X(0) = x_0$$
$$\dot{X}(0) = -\frac{2\sqrt{s}\nabla f(x_0)}{1+\sqrt{\mu s}}$$

By following a similar derivation the high-resolution differential equation for the *heavy-ball* method can be shown to be:

$$\ddot{X}(t) + 2\sqrt{\mu}\dot{X}(t) + (1+\sqrt{\mu s})\nabla f(X(t)) = 0 \tag{5.3}$$

with $X(0) = x_0$ and $\dot{X}(0) = -\frac{2\sqrt{s}\nabla f(x_0)}{1+\sqrt{\mu s}}$.

Analogously, the *high-resolution* ODE for Nesterov's Accelerated Gradient for convex functions can be shown to be:

$$\ddot{X}(t) + \frac{3}{t}\dot{X}(t) + \sqrt{s}\nabla^2 f(X(t))\dot{X}(t) + (1 + \frac{3\sqrt{s}}{2t})\nabla f(X(t)) = 0 \qquad (5.4)$$

for $t \geq 3\sqrt{s}/2$, with $X(3\sqrt{s}/2) = x_0$ and $\dot{X}(3\sqrt{s}/2) = -\sqrt{s}\nabla f(x_0)$.

Remark 3. *Note that all the high-resolution differential equations reduce to their low-resolution counterparts introduced in chapter 4 in the limit $s \to 0$, as expected.*

Remark 4. *From the derivation we have made for 5.2, the contribution of the gradient correction term in the high-resolution ODE is $\sqrt{s}\nabla^2 f(X)\dot{X}$. Hence, viewing the coefficient of $\dot{X}$ as a damping ratio, the coefficient $2\sqrt{\mu} + \sqrt{s}\nabla^2 f(X)$ of $\dot{X}$ in the high-resolution ODE 5.2 is adaptive of the position X, in contrast to the fixed damping ratio $2\sqrt{\mu}$ for the heavy-ball high-resolution ODE 5.3. To appreciate the effect of this adaptivity, suppose that $\dot{X}$ is highly correlated with an eigenvector of $\nabla^2 f(X)$ with a large eigenvalue (and therefore, if we follow this direction we expect a big decrease in the objective function). Then, the friction increases and decelerates the trajectory of the solution of 5.2. This property is desirable because a small step in the presence of high curvature (directions correlated with large eigenvalues of the Hessian) generally avoids big oscillations. This lack of big oscillations is one of the differences that can be observed experimentally between Polyak's heavy-ball and Nesterov's Accelerated Gradient and that gives the latter an advantage for certain objective functions.*

5.2 Convergence: the Continuous Case

Now that we have different *high-resolution* differential equations for the different optimization methods, we will state a set of theorems that characterize their convergence properties and how to translate them to the discrete case. We will not state existence and uniqueness results for the *high-resolution* differential equations presented above. From now on, we will assume that we have existence and uniqueness of solutions for all of them under the conditions stated in the theorems that we present next:

Theorem 14 ([5]). *(Convergence of 5.2). Let $f \in \mathscr{S}^2_{\mu,L}(\mathbb{R}^n)$. For any step size $0 \leq s \leq 1/L$, the solution $X = X(t)$ of the high-resolution ODE 5.2 satisfies:*

$$f(X(t)) - f(x_*) \leq \frac{2\left\|x_0 - x_*\right\|^2}{s} e^{-\frac{\sqrt{\mu}t}{4}}$$

The next lemma states the key property used in the proof of the previous theorem:

Lemma 15. *(Lyapunov function for 5.2) Let $f \in \mathscr{S}^2_{\mu,L}(\mathbb{R}^n)$. For any step size $s > 0$, and with $X = X(t)$ being the solution to 5.2, the Lyapunov function*

$$V(t) = (1 + \sqrt{\mu s})(f(X) - f(x_*) + \frac{1}{4}\left\|\dot{X}\right\|^2 + \frac{1}{4}\left\|\dot{X} + 2\sqrt{\mu}(X - x_*) + \sqrt{s}\nabla f(X)\right\|^2$$

satisfies

$$\frac{dV(t)}{dt} \leq -\frac{\sqrt{\mu}}{4}V(t) - \frac{\sqrt{s}}{2}\Big[\left\|\nabla f(X(t))\right\|^2 + \dot{X}(t)^T\nabla^2 f(X(t))\dot{X}(t)\Big]$$

Note that in particular, $\frac{dV(t)}{dt} < 0$, because f is convex and therefore the Hessian is positive definite.
Now we state the same type of result for the *high-resolution* Heavy-Ball ODE:

Theorem 16 ([5]). *(Convergence of 5.3)*
Let $f \in \mathscr{S}^2_{\mu,L}(\mathbb{R}^n)$. For any step size $0 < s \leq 1/L$, the solution $X = X(t)$ of the high-resolution ODE 5.3 satisfies

$$f(X(t)) - f(x_*) \leq \frac{7\left\|x_0 - x_*\right\|^2}{2s} e^{-\frac{\sqrt{\mu}t}{4}}$$

Just like before, the result is based on the following key lemma:

Lemma 17. *(Lyapunov function for 5.3) Let $f \in \mathscr{S}^2_{\mu,L}(\mathbb{R}^n)$. For any step size $s > 0$, the Lyapunov function*

$$V(t) = (1 + \sqrt{\mu s})(f(X) - f(x_*) + \frac{1}{4}\left\|\dot{X}\right\|^2 + \frac{1}{4}\left\|\dot{X} + 2\sqrt{\mu}(X - x_*)\right\|^2$$

satisfies:

$$\frac{dV(t)}{dt} \leq -\frac{\sqrt{\mu}}{4}V(t)$$

5.3 Convergence: the Discrete Case

In this section we rewrite the results from the previous section in the discrete case. This will be done by discretizing the Lyapunov functions derived in 5.2. This discretization can often be tricky, and the details can be found at [5]. Overall, we are able to derive convergence results we already knew from 2, but which will now be proven via a Lyapunov perspective.

Theorem 18 ([5])**.** *(Convergence of Nesterov's Accelerated Gradient method for strongly-convex functions) Let $f \in \mathscr{S}^2_{\mu,L}(\mathbb{R}^n)$. If the step size is set to $s = \frac{1}{4L}$, the iterates $\{x_k\}_{k=0}^{\infty}$ generated by Nesterov's Accelerated Gradient method for strongly-convex functions satisfy:*

$$f(x_k) - f(x_*) \leq \frac{5L\left\|x_0 - x_*\right\|^2}{1 + \frac{1}{12}\sqrt{\mu/L}}$$

for all $k \geq 0$

Theorem 18 is proven by using a result that relates the rate of decay of the discrete-time Lyapunov function (recall that the derivative in discrete time translates to the difference of the value of the function between two timesteps, $V(k+1) - V(k)$.

Lemma 19 ([5])**.** *Let $f \in \mathscr{S}^2_{\mu,L}(\mathbb{R}^n)$. Taking any step size $0 < s \leq \frac{1}{4L}$, the discrete Lyapunov function*

$$V(k) = \frac{1+\sqrt{\mu s}}{1-\sqrt{\mu s}}(f(x_k) - f(x_*)) + \frac{1}{4}\left\|v_k\right\|^2 + \frac{1}{4}\left\|v_k + \frac{2\sqrt{\mu}}{1-\sqrt{\mu s}}(x_{k+1} - x_*) + \sqrt{s}\nabla f(x_k)\right\|^2 - \frac{s\left\|\nabla f(x_k)\right\|^2}{2(1-\sqrt{\mu s})}$$

satisfies:

$$V(k+1) - V(k) \leq -\frac{\sqrt{\mu s}}{6}V(k+1)$$

Now we state the same results for the *heavy-ball* method.

Theorem 20 ([5]). *(Convergence of the heavy-ball method). Let $f \in \mathscr{S}^2_{\mu,L}(\mathbb{R}^n)$. If the step size is set to $s = \frac{\mu}{16L^2}$, the iterates $\{x_k\}_{k=0}^{\infty}$ generated by the heavy-ball method satisfy:*

$$f(x_k) - f(x_0) \leq \frac{5L \left\|x_0 - x_*\right\|^2}{(1 + \frac{\mu}{16L})^k}$$

for all $k \geq 0$

The theorem relies on this lemma:

Lemma 21. *Let $f \in \mathscr{S}^2_{\mu,L}(\mathbb{R}^n)$. For any step size $s > 0$, the discrete Lyapunov function*

$$V(k) = \frac{1 + \sqrt{\mu s}}{1 - \sqrt{\mu s}}(f(x_k) - f(x_*)) + \frac{1}{4}\left\|v_k\right\|^2 + \frac{1}{4}\left\|v_k + \frac{2\sqrt{\mu}}{1 - \sqrt{\mu s}}\right\|^2$$

satisfies

$$\begin{aligned} V(k+1) - V(k) \leq & -\sqrt{\mu s}\min\{\frac{1 - \sqrt{\mu s}}{1 + \sqrt{\mu s}}, \frac{1}{4}\}V(k+1) \\ & - \Big[\frac{3\sqrt{\mu s}}{4}(\frac{1 + \sqrt{\mu s}}{1 - \sqrt{\mu s}})(f(x_{k+1}) - f(x_*)) - \frac{s}{2}(\frac{1 + \sqrt{\mu s}}{1 - \sqrt{\mu s}})^2 \left\|\nabla f(x_{k+1})\right\|^2\Big] \end{aligned}$$

Chapter 6

Discretizing High Resolution Differential Equations: an event-triggered approach

In this chapter we are gong to use the *high-resolution* differential equations introduced in the Chapter 5 to find new discrete-time optimization algorithms. To do so, we are going to discretize these differential equations by using event-triggered control ideas introduced in Chapter 3.
It must be said beforehand that since high-resolution differential equations were introduced in [5], a number of works have also explored the discretization of accelerated continuous models. The work [1] shows that the forward Euler method can be inefficient and even become unstable after a few iterations. In [8], it is shown that high-order Runge-Kutta integrators can also be used to retain acceleration when discretizing the low-resolution differential equation for Nesterov's method for convex functions. The paper [4] analyzes the properties of explicit, implicit and symplectic integrators for the high-resolution differential equations for the *heavy-ball* method and NAG for strongly-convex functions.
The work presented in this chapter is mainly taken from [16].

6.1 Forward-Euler Discretization of Dynamical Systems via State-Triggered Control

Consider a dynamical system on $\mathbb{R}^n$,

$$\dot{p} = Y(p) \tag{6.1}$$

where $Y : \mathbb{R}^n \to \mathbb{R}^n$. Assume p_* is a globally asymptotically stable equilibrium point under this dynamics, and a Lyapunov function $V : \mathbb{R}^n \to \mathbb{R}$ is available to guarantee it. Assume V decreases at a rate according to:

$$\dot{V} = \langle \nabla V(p), Y(p) \rangle \leq -\alpha V(p)$$

for all $p \in \mathbb{R}^n$. Obviously the system 6.1 doesn't have a control, but we will apply the same ideas discussed in Chapter 3. Instead of sampling the control and keeping it constant until a triggering condition is violated, we will sample the state and keep it constant until a triggering condition is violated. Consider the sampled implementation of 6.1 given by:

$$\dot{p} = Y(\hat{p}) \tag{6.2}$$

with $p(0) = \hat{p}$. Solving the differential equation:

$$p(t) = \hat{p} + tY(\hat{p})$$

Note that this can be though as the Forward (or Explicit) Euler discretization of 6.1 with stepsize t. This is an interesting observation on its own: variable-stepsize Euler discretizations are equivalent to the State-Triggered implementation of a dynamical system.

To find the time we have to do the next sampling (or equivalently, the stepsize), assume we have access to a continuous function $g : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ that satisfies $g(p, 0) < 0$ for all $p \in \mathbb{R}^n \backslash \{p_*\}$ and

$$\dot{V}(p(t)) + \alpha V(p(t)) \leq g(\hat{p}, t) \tag{6.3}$$

is satisfied along the solutions of 6.2. Then, for each $i \in \mathbb{N}$, the dynamics are given by

$$\dot{p} = Y(p_i), \ p(0) = p_i \tag{6.4}$$
$$p(t) = p_i + tY(p_i) \tag{6.5}$$

and the next triggering time can be determined by:

$$t_{i+1} = \min\{t | t > t_i \text{ such that } g(p_i, t) = 0\} \tag{6.6}$$

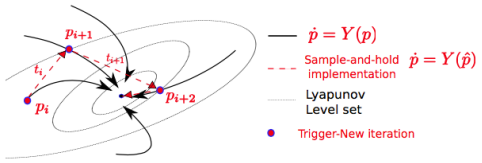

Figure 6.1: Equivalence between an state-triggered implementation and variable-stepsize explicit Euler. The black lines correspond to the trajectories of the original dynamics 7.10. The red lines correspond to the trajectories of the sampled implementation 6.2

Note that overall, 6.4 can be thought of as a *hybrid* system, i.e, a dynamical system where the vector field describing its dynamics changes from time to time (in our case, this change occurs when the triggering condition is violated).
Note that by construction we have $\dot{V}(p(t)) \leq -\alpha V(p(t))$ along the dynamics 6.4. Just like we mentioned in Chapter 3, if g is such that t_{i+1} can be determined explicitly only with knowledge of p_i and it does not require the continuous monitoring of p, one refers to this design as *self-triggered*.

6.2 Triggered Discretization of the Heavy Ball Continuous Model

Here we will derive a discretization, using the methodology described in the previous section, of the *high-resolution* differential equation for the *heavy-ball* method 5.3. A similar discussion can be made for the rest of high-resolution differential equations presented in chapter 5, but will not be included here due to space constrains. We refer the interested reader to the supplementary material from [16].

Let $f \in \mathscr{S}^1_{\mu,L}$. We restart by rewriting the recond order differential equation 5.3 as a system of first order differential equations:

$$\begin{bmatrix}\dot{x}\\ \dot{v}\end{bmatrix} = \begin{bmatrix} v \\ -2\sqrt{\mu}v - (1+\sqrt{\mu s})\nabla f(x))\end{bmatrix}, \tag{6.7}$$

with initial conditions $x(0) = x_0$, $v(0) = -\frac{2\sqrt{s}\nabla f(x_0)}{1+\sqrt{\mu s}}$. We refer to this dynamics as X_{hb}.

We know from theorem 16 that there exists a Lyapunov function V positive definite with respect to $[x_*, 0]$ satisfying $\dot{V}(p(t)) \leq -\frac{\sqrt{\mu}}{4}V(p(t))$ along the dynamics 6.7. As a consequence, $[x_*, 0]$ is globally asymptotically stable. Thus, we have all the necessary ingredients to develop a state-triggered discretization that preserves the convergence rate, like the one described in the previous section.
The only obstacle is the fact that the Lyapunov function depends on the optimizer, which in an optimization problem is unknown, so we want to find a bounding function $g(\hat{p}, t)$ that doesn't depend on the optimizer and that satisfies:

$$\dot{V}(p(t)) + \alpha V(p(t)) \leq g(\hat{p}, t) \tag{6.8}$$
$$g(\hat{p}, 0) < 0 \tag{6.9}$$

We will do so by using the convexity properties of the function f. The next proposition defines such a g

Proposition 22 ([16])**.** *For the sample-and-hold dynamics $\dot{p} = X_{hb}(\hat{p})$, $p(0) = \hat{p}$, $s \geq 0$ and $0 \leq \alpha \leq \sqrt{\mu}/4$, let*

$$\begin{aligned}
&\tfrac{d}{dt}V(p(t)) + \alpha V(p(t)) \\
&= \langle \nabla V(\hat{p} + tX_{hb}(\hat{p})), X_{hb}(\hat{p})\rangle + \alpha V(\hat{p} + tX_{hb}(\hat{p})) \\
&= \underbrace{\langle \nabla V(\hat{p} + tX_{hb}(\hat{p})) - \nabla V(\hat{p}), X_{hb}(\hat{p})\rangle}_{I} \\
&+ \underbrace{\alpha(V(\hat{p} + tX_{hb}(\hat{p})) - V(\hat{p}))}_{II} + \underbrace{\langle \nabla V(\hat{p}), X_{hb}(\hat{p})\rangle + \alpha V(\hat{p})}_{III}.
\end{aligned}$$

Then, the following bounds hold:

1. *Term $I \leq A_{ET}(\hat{p}) \leq A_{ST}(\hat{p})t$;*

2. *Term* $II \leq BC_{ET}(\hat{p}, t) \leq B_{ST}(\hat{p})t + C_{ST}(\hat{p})t^2$;

3. *Term* $III \leq D_{ET}(\hat{p}, t) = D_{ST}(\hat{p})$,

where $\hat{p} = [\hat{x}, \hat{v}]$ *and*

$$\begin{aligned}
A_{ET}(\hat{p}, t) &= (1 + \sqrt{\mu s})\langle \nabla f(\hat{x} + t\hat{v}) - \nabla f(\hat{x}), \hat{v}\rangle + 2\mu t \left\|\hat{v}\right\|^2 \\
&\quad + 2\sqrt{\mu}(1 + \sqrt{\mu s})\langle \nabla f(\hat{x}), \hat{v}\rangle + t(1 + \sqrt{\mu s})^2 \left\|\nabla f(\hat{x})\right\|^2 \\
BC_{ET}(\hat{p}, t) &= \alpha(1 + \sqrt{\mu s})(f(\hat{x} + t\hat{v}) - f(\hat{x})) + t^2\frac{\alpha}{4}\left\|-2\sqrt{\mu}\hat{v} - (1 + \sqrt{\mu s})\nabla f(\hat{x})\right\|^2 \\
&\quad - \alpha t(1 + \sqrt{\mu s})\langle \hat{v}, \nabla f(\hat{x})\rangle - t\sqrt{\mu}\left\|\hat{v}\right\|^2 \\
&\quad + t^2\frac{\alpha}{4}\left\|(1 + \sqrt{\mu s})\nabla f(\hat{x})\right\|^2 - \alpha t\sqrt{\mu}(1 + \sqrt{\mu s})\left\|\nabla f(\hat{x})\right\|^2 / L \\
D_{ET}(\hat{p}) &= (\frac{3\alpha}{4} - \sqrt{\mu})\left\|v\right\|^2 + \\
&\quad ((1 + \sqrt{\mu s})\frac{\alpha - \sqrt{\mu}}{2L} + (\alpha 2\mu - \frac{\sqrt{\mu}(1 + \sqrt{\mu s})\mu}{2})\frac{1}{L^2})\left\|\nabla f(\hat{x})\right\|^2 \\
A_{ST}(\hat{p}) &= (1 + \sqrt{\mu s})L\left\|\hat{v}\right\|^2 + 2\sqrt{\mu}(1 + \sqrt{\mu s})\langle \nabla f(\hat{x}), \hat{v}\rangle + \\
&\quad 2\mu\left\|\hat{v}\right\|^2 + (1 + \sqrt{\mu s})^2\left\|\nabla f(\hat{x})\right\|^2 \\
B_{ST}(\hat{p}) &= -\alpha\sqrt{\mu}\left\|\hat{v}\right\|^2 - \frac{\alpha\sqrt{\mu}(1 + \sqrt{\mu s})}{L}\left\|\nabla f(\hat{x})\right\|^2 \\
C_{ST}(\hat{p}) &= \alpha(1 + \sqrt{\mu s})\frac{L}{2}\left\|\hat{v}\right\|^2 + \frac{\alpha}{4}\left\|-2\sqrt{\mu}\hat{v} - (1 + \sqrt{\mu s})\nabla f(\hat{x})\right\|^2 \\
&\quad + \frac{\alpha}{4}(1 + \sqrt{\mu s})^2\left\|\nabla f(\hat{x})\right\|^2
\end{aligned}$$

The proof of this proposition can be found at the supplementary material for [16].

Now we can define:

$$\begin{aligned}
g_{ET} &= A_{ET}(\hat{p}, t) + BC_{ET}(\hat{p}, t) + D_{ET}(\hat{p}, t) \\
g_{ST} &= C_{ST}(\hat{p})t^2 + (A_{ST}(\hat{p}) + B_{ST}(\hat{p}))t + D_{ST}(\hat{p})
\end{aligned}$$

With these functions defined and from Proposition 22 the following bounds hold:

$$\frac{d}{dt}V(p(t)) + \alpha V(p(t)) \leq g_{ET}(\hat{p}, t) \leq g_{ST}(\hat{p}, t) \tag{6.10}$$

Now the stepsize starting from $\hat{p}$ is calculated by:

$$\text{step}_{\#}(\hat{p}) = \min_t\{t > 0 \text{ such that } g_{\#}(\hat{p}, t) = 0\} \tag{6.11}$$

where $\# \in \{ET, ST\}$.

Note that $g_{ET}(\hat{p}, t) = 0$ is an implicit equation on t (hence the subscript, *event-triggered*). Instead $g_{ST}(\hat{p}, t)$ is a quadratic equation and can thus be solved explicitly with knowledge only of the current state $\hat{p}$ (hence the subscript, *self-triggered*). Moreover, since $D_{ST}(\hat{p}) < 0$ when $\alpha \leq \frac{\sqrt{\mu}}{4}$, there is always only one positive solution given by:

$$step_{ST}(\hat{p}) = \frac{-(A_{ST}(\hat{p}) + B_{ST}(\hat{p}) + \sqrt{(A_{ST}(\hat{p}) + B_{ST}(\hat{p}))^2 - 4C_{ST}(\hat{p})D_{ST}(\hat{p})})}{2C_{ST}(\hat{p})} \tag{6.12}$$

The resulting state-triggered algorithm is given as follows:

Algorithm 3: Triggered Forward-Euler algorithm

Initialization: Initial point (p_0), convergence rate (α), objective function (f), tolerance (ϵ) ;
Set: $k = 0$;
while $\|\nabla f(x)\| \geq \epsilon$ **do**
  Compute stepsize t_k at current point according to 6.11 ;
  Compute next iterate $p_{k+1} = p_k + t_k X_{hb}(p_k)$;
  Set $k = k + 1$
end

The MATLAB code for algorithm 3 is included in Appendix C. The following theorem states that this algorithm satisfies the two conditions that any event-triggered algorithm must satisfy: non-Zeno behavior and desired decay of the Lyapunov function:

Theorem 23 ([16]). *For $0 \leq \alpha \leq \sqrt{\mu}/4$ and $\# \in \{ET, ST\}$, Algorithm 3 is a variable-stepsize integrator with the following properties:*

(i) the stepsize is uniformly lower bounded by a positive constant. Namely

$$-\bar{c_2} + \sqrt{\bar{c_2}^2 + \bar{c_1}} \leq step_{ST}(p)$$

where

$$\bar{c_1} = \min\{\frac{2(\sqrt{\mu} - \frac{3\alpha}{4})}{\alpha(4\mu + L\sqrt{\mu s} + L)}, \frac{2(-4\alpha\mu + L(\sqrt{\mu} - \alpha)(\sqrt{\mu s} + 1) + \mu^{3/2}(\sqrt{\mu s} + 1))}{3\alpha L^2(\sqrt{\mu s} + 1)^2}\}$$

$$\bar{c_2} = \max\{\frac{(2\mu + \sqrt{\mu} + L)(\sqrt{\mu s} + 1)}{\alpha(4\mu + L\sqrt{\mu s} + L)}, \frac{2(\sqrt{\mu} + \sqrt{\mu s} + 1)}{3\alpha(\sqrt{\mu s} + 1)}\}$$

(ii) $\frac{d}{dt}V(p_k + tX_{hb}(p_k) \leq -\alpha V(p_k + tX_{hb}(p_k))$ *for all* $t \in [0, t_k]$ *and all* $k \in \{0\} \cup \mathbb{N}$

As a consequence,

$$f(x_{k+1}) - f(x_*) \leq \frac{7\left\|x(0) - x_*\right\|^2}{2s} e^{-\alpha \sum_{i=0}^{k} t_i}$$

for all $k \in \{0\} \cup \mathbb{N}$

Proof. Since $g_{ET}(p,t) \leq g_{ST}(p,t)$ we have $step_{ST}(p) \leq step_{ET}(p)$ and therefore it is enough to prove the first claim for the *self-triggered* case. We start by rewriting 6.12 like this:

$$\text{step}_{ST}(p) = -\frac{A_{ST}(p) + B_{ST}(p)}{2C_{ST}(p)} + \sqrt{\Big(\frac{A_{ST}(p) + B_{ST}(p)}{2C_{ST}(p)}\Big)^2 - \frac{D_{ST}(p)}{C_{ST}(p)}}$$

We will start by showing that $\frac{D_{ST}(p)}{C_{ST}(p)}$ is uniformly lower bounded in p. Bound, using $\left\|a+b\right\|^2 \leq 2\left\|a\right\|^2 + 2\left\|b\right\|^2$:

$$C_{ST}(p) \leq \alpha\Big((1 + \sqrt{\mu s})\frac{L}{2} + 2\mu\Big)\left\|v\right\|^2 + \alpha\frac{3}{4}(1 + \sqrt{\mu s})^2\left\|\nabla f(x)\right\|^2$$

and therefore

$$\frac{-D_{ST}(p)}{C_{ST}(p)} \geq \frac{(\frac{3\alpha}{4} - \sqrt{\mu})\left\|v\right\|^2 + \Big((1 + \sqrt{\mu s})\frac{\alpha - \sqrt{\mu}}{2L} + (\alpha 2\mu - \frac{\sqrt{\mu}(1+\sqrt{\mu s})\mu}{2})\frac{1}{L^2}\Big)\left\|\nabla f(\hat{x})\right\|^2}{\alpha\Big((1 + \sqrt{\mu s})\frac{L}{2} + 2\mu\Big)\left\|v\right\|^2 + \alpha\frac{3}{4}(1 + \sqrt{\mu s})^2\left\|\nabla f(x)\right\|^2}$$

By renaming $\|\nabla f(x)\| = z_1$ and $\|v\| = z_2$ the last expression takes the form

$$\frac{\beta_1 z_1^2 + \beta_2 z_2^2}{\beta_3 z_1^2 + \beta_4 z_2^2} \tag{6.13}$$

It can be shown by elementary calculus that this function is upper and lower bounded (a proof of this result can be found at the supplementary material of [16]). I.e, there exist positive constants c_1 and c_2 such that:

$$0 < c_1 \leq \frac{\beta_1 z_1^2 + \beta_2 z_2^2}{\beta_3 z_1^2 + \beta_4 z_2^2} \leq c_2 \quad \text{for all } z_1, z_2 \in \mathbb{R}\backslash\{0\} \tag{6.14}$$

Thus,

$$\frac{-(A_{ST}(p) + B_{ST}(p))}{2C_{ST}(p)} + \sqrt{(\frac{A_{ST}(p) + B_{ST}(p)}{2C_{ST}(p)})^2 + c_1} \leq \text{step}_{ST}(p)$$

Now it is easy to see that the function $f(z) = -z + \sqrt{z^2 + c_1}$ is monotonically decreasing and positive everywhere. If we can prove that $z = \frac{A_{ST}(p)+B_{ST}(p)}{2C_{ST}}$ is upper bounded, then $f(z)$ is lower bounded by a positive constant. To achieve it, let's use:

$$C_{ST}(p) \geq \alpha\Big((1 + \sqrt{\mu s})\frac{L}{2}\|v\|^2 + \frac{(1 + \sqrt{\mu s})^2}{4}\|\nabla f(x)\|^2\Big)$$

and

$$\begin{aligned} A_{ST}(p) + B_{ST}(p) \leq A_{ST}(p) \leq (1 + \sqrt{\mu s})L\|v\|^2 + \sqrt{\mu}(1 + \sqrt{\mu s})\|\nabla f(x)\|^2 \\ \sqrt{\mu}(1 + \sqrt{\mu s})\|v\|^2 + 2\mu\|v\|^2 + (1 + \sqrt{\mu s})^2\|\nabla f(x)\|^2 \end{aligned}$$

where we have used Cauchy-Schwartz

$$\langle a, b\rangle \leq \|a\|\,\|b\|$$

and Young's inequality

$$ab \leq \sqrt{\frac{a^2 + b^2}{2}}$$

for the last estimate. Now, the fraction $\frac{A_{ST}(p)+B_{ST}(p)}{2C_{ST}}$ has the form 6.13 so it is upper bounded.
It should be noted that the values $\bar{c_1}$ and $\bar{c_2}$ given in the statement of the theorem can be calculated by using the explicit values for c_1 and c_2 in 6.14 given in the supplementary material for [16]. The second part of the statement of the theorem follows by construction. □

Corollary 24. *Let the stepsize t_k be lower bounded by $\hat{t}$ for all $k \in \mathbb{N}$. Then,*

$$f(x_{k+1}) - f(x_*) \in \mathcal{O}(exp(-\frac{\sqrt{\mu}}{4}\hat{t})^k)$$

Proof. Follows immediately from the second part of theorem 23 □

Even though we don't observe acceleration in simulations like 6.4, i.e, $\exp(-\frac{\sqrt{\mu}}{4}\hat{t}) > 1 - \frac{\sqrt{\mu}}{L}$, this might be because of the simplicity of the integrator used. In Chapter 7 we will suggest some modifications on 3 that lead to improved convergence rates.

It should be noted that the same triggered discretizations can be applied to the high-resolution differential equation for Nesterov's Accelerated Gradient.

$$\begin{bmatrix}\dot{x}\\ \dot{v}\end{bmatrix} = \begin{bmatrix} v \\ -2\sqrt{\mu}v - \sqrt{s}\nabla^2 f(x)v - (1+\sqrt{\mu s})\nabla f(x))\end{bmatrix}, \tag{6.15}$$

with initial conditions $x(0) = x_0$, $v_0 = -\frac{2\sqrt{s}\nabla f(x_0)}{1+\sqrt{\mu s}}$. A detailed description of a self-triggered algorithm for 6.15 can be found in the supplementary material for [16]. Furthermore, the authors show in a simulation that the triggered discretizations might be better than the symplectic and implicit Euler discretizations derived in [4].

6.2.1 Simulations

The next simulation shows how the Event-Triggered implementation is more accurate than the Self-Triggered Implementation for a fairly well-conditioned function:

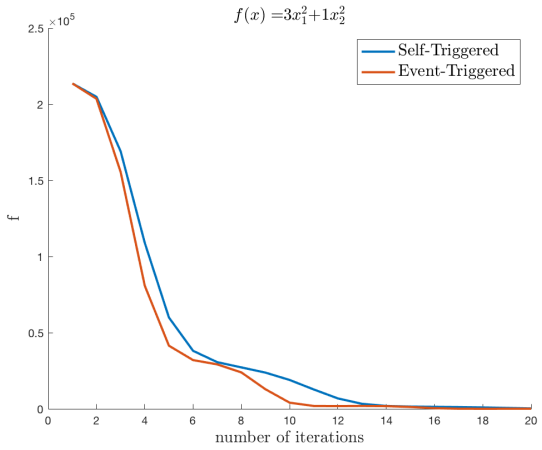

Figure 6.2: Value of $f(x, y) = 3x^2 + y^2$ in terms of the number of iterations for the event-triggered version of 3 and the self-triggered version of 3 with initial conditions $x_0 = [1121, 2333]$ and $v_0 = -0.0456\nabla f(x_0)$

For more ill-conditioned objectives the two approaches tend to be very similar, as the next simulation shows:

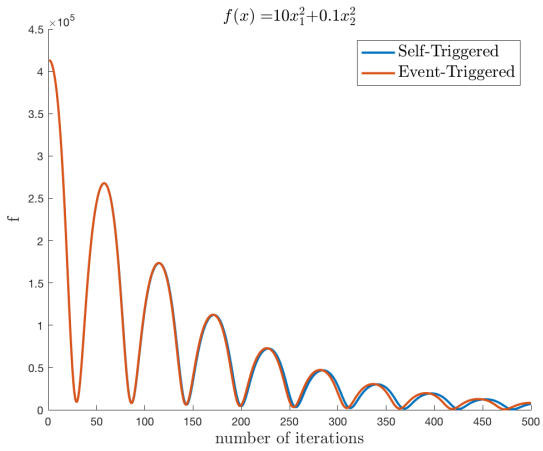

Figure 6.3: Value of $f(x, y) = 10x^2 + 0.1y^2$ in terms of the number of iterations for the event-triggered version of 3 and the self-triggered version of 3 with initial conditions $x_0 = [201, 304]$ and $v_0 = -0.0456\nabla f(x_0)$

The following simulation shows that algorithm 3 doesn't seem to achieve

acceleration.

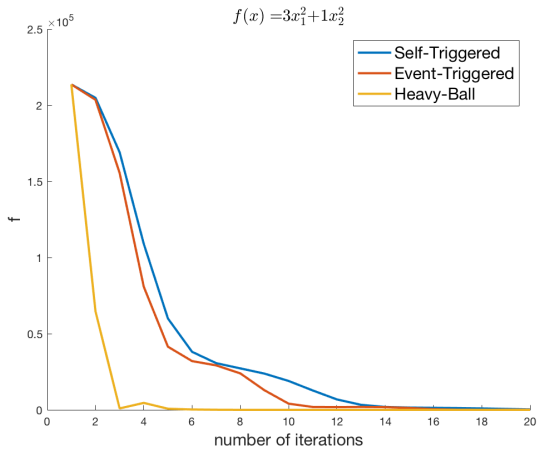

Figure 6.4: Value of $f(x, y) = 3x^2 + y^2$ in terms of the number of iterations for the event-triggered version of algorithm 3 and the self-triggered version of algortihm 3 and Polyak's Heavy-Ball method

Chapter 7

Some Improvements on Event-Triggered Algorithms

This chapter builds on the discretization method introduced in Chapter 6. Namely, we will show how to improve it in three different (but potentially compatible) directions:

(i) Different triggering conditions, to which section 7.1 is dedicated,

(ii) Alternative sampling procedures, which will be explored in section 7.2,

(iii) Considering a more exact dynamics, in 7.4

At this point it should be stated that proving theoretically that any of the algorithms that will be presented below are strictly better (i.e, have a strictly better convergence rate) than algorithm 3 is a tough problem, because even though for the same initial conditions one of the algorithms might take a provably longer step (which might seem to lead to a better convergence rate), the new initial conditions that result from taking a more cautious step might be benefitial in the future. In other words, by the time this thesis is being written, we don't know whether a *greedy* strategy like the one followed in 3 (where we try to take the **longest** possible step before violating the triggering condition) is the optimal strategy for triggered discretization algorithms.

7.1 Performance-Based Trigger

In Chapter 6 we used what is known as the *derivative-based* trigger. This means that the condition

$$\dot{V} + \alpha V \leq 0 \tag{7.1}$$

is satisfied by the triggered dynamics (with $\alpha \in [0, \sqrt{\mu}/4]$) and the triggering condition is defined by an upper bound of 7.1.

It's easy to see by the Comparison Lemma that condition 7.1 implies:

$$V(p(t)) \leq e^{-\alpha t} V(p(0)) \tag{7.2}$$

which can also be used as a triggering conditon by itself and is known as the *performance-based* trigger.

Remark 5. *If we construct an even-triggered algorithm that satisfies 7.2, we get a convergence rate on the objective function given by:*

$$f(x(t)) - f(x_*) \leq V(p(t)) \leq e^{-\alpha t} V(p(0)) = \mathcal{O}(e^{-\alpha t})$$

and applying the same inequality piece-wise, we get:

$$f(x_{k+1}) - f(x_*) = \mathcal{O}(e^{-\alpha \sum_{i=0}^{k} t_i})$$

Remark 6. *The performance-based triggering condition is weaker than the derivative-based one, which means that the performance-based trigger might lead to better discretizations, because it might provide larger stepsizes and still maintain the required convergence rates.*

We will now derive a series of conditions which are equivalent to the *performance-based* trigger.

$$\begin{aligned} & V(p(t)) \leq e^{-\alpha t} V(p(0)) \\ \Leftrightarrow\ & V(p(t)) - e^{-\alpha t} V(p(0)) \leq 0 \\ \Leftrightarrow\ & e^{\alpha t} V(p(t)) - V(p(0)) \leq 0. \end{aligned}$$

Writing the first addend as the integral of its derivative plus its initial value, we get:

$$\begin{aligned}
&\int_0^t \tfrac{d}{d\tau}(e^{\alpha\tau}V(p(\tau)))d\tau + e^{0t}V(p(0)) - V(p(0)) \\
&= \int_0^t e^{\alpha\tau}\alpha V(p(\tau)) + e^{\alpha t}\langle\nabla V(p(\tau)), X_{\text{hb}}(p(\tau))\rangle d\tau \\
&= \int_0^t e^{\alpha\tau}(\underbrace{\langle\nabla V(p(\tau)), X_{\text{hb}}(p(\tau))\rangle + \alpha V(p(\tau))}_{\text{Derivative-based}})\, d\tau.
\end{aligned} \tag{7.3}$$

Once again we have the same problem as when we tried to evaluate the *derivative-based* condition: the triggering condition depends on the optimizer x_*, which is unknown, so we will try to find an upper bound which is independent of x_*. In this case, we can use the upper bound for the *derivative-based* condition derived in chapter 6:

$$\int_0^t e^{\alpha\tau}(\underbrace{\langle\nabla V(p(\tau)), X_{\text{hb}}(p(\tau))\rangle + \alpha V(p(\tau))}_{\text{Derivative-based}})\, d\tau \leq \int_0^t e^{\alpha\tau} g_{ET}(\hat{p},\tau) \leq \int_0^t e^{\alpha\tau} g_{ST}(\hat{p},\tau)$$

where $g_{ET}(\hat{p},t)$ and $g_{ST}(\hat{p},t)$ are the ones defined in Chapter 6.

Since $g_{ST}(p,t)$ is a second-order polynomial, we can compute the integral $\int_0^t e^{\alpha\tau} g_{ST}(\hat{p},\tau)d\tau$ explicitly:

$$\begin{aligned}
&\int_0^t e^{\alpha\tau}(a\tau^2 + b\tau + c)d\tau \\
&= [e^{\alpha t}(\tfrac{a}{\alpha}t^2 + (\tfrac{b}{\alpha} - \tfrac{2a}{\alpha^2})t + \tfrac{c}{\alpha} - \tfrac{b}{\alpha^2} + \tfrac{2a}{\alpha^3})]_0^t \\
&= e^{\alpha t}(\tfrac{a}{\alpha}t^2 + (\tfrac{b}{\alpha} - \tfrac{2a}{\alpha^2})t + \tfrac{c}{\alpha} - \tfrac{b}{\alpha^2} + \tfrac{2a}{\alpha^3}) - (\tfrac{c}{\alpha} - \tfrac{b}{\alpha^2} + \tfrac{2a}{\alpha^3})
\end{aligned}$$

The formula for the next time we need to update the control is given by:

$$t_{i+1} = \min\{t \mid t > t_i \text{ such that } \int_0^t e^{\alpha t} g_{\#}(p,t) = 0\}.$$

where $\# \in \{\text{ET}, \text{ST}\}$. Which leads to the obvious definition of the stepsize,

$$\text{step}_{\#}^{P}(p) = \min_t\{\int_0^t e^{\alpha\tau} g_{\#}(p,\tau)d\tau = 0\} \tag{7.4}$$

where $\# \in \{\text{ET}, \text{ST}\}$. Finally, we introduce the corresponding algorithm.

Algorithm 4: Performance-Based Algorithm

Initialization: Initial point (p_0), convergence rate (α), objective function (f), tolerance (ϵ);
Set: $k = 0$;
while $\|\nabla f(x)\| \geq \epsilon$ **do**
    Compute stepsize t_k according to (7.4);
    Compute next iterate $p_{k+1} = p_k + t_k X_{\text{hb}}(p_k)$;
    Set $k = k + 1$
end

Next we state the equivalent of Theorem 23 of Chapter 6 for Algorithm 4:

Theorem 25. *For $0 \leq \alpha \leq \frac{\sqrt{\mu}}{4}$ and $\# \in \{\text{ET}, \text{ST}\}$, Algorithm 4 is a variable stepsize integrator with the following properties*

(i) the stepsize is uniformly lower bounded by a positive constant;

(ii) $V(p_{k+1}) \leq e^{-\alpha t} V(p_k)$ for all $k \in \{0\} \cup \mathbb{N}$.

As a consequence, it follows that $f(x_{k+1}) - f(x_*) = \mathcal{O}(e^{-\alpha \sum_{i=0}^{k} t_i})$.

Proof. We show that $\text{step}_{\text{ST}}^{P}(p)$ is lower bounded and the remaining results follow easily (because $\text{step}_{\text{ET}}^{P}(p)$ is lower bounded by $\text{step}_{\text{ST}}^{P}(p)$). Our argument relies on the observation that $\text{step}_{\text{ST}}(p) \leq \text{step}_{SP}^{P}(p)$, where $\text{step}_{\text{ST}}(p)$ is the step taken in the *self-triggered* case of algorithm 3. Recall that this stepsize is defined by

$$\text{step}_{\text{ST}}(p) = \min_{t} \{g_{\text{ST}}(p, t) = 0\}.$$

In Theorem 23 it has been shown that step_{ST} is positively lower bounded. To see $\text{step}_{\text{ST}}(p) \leq \text{step}_{SP}^{P}(p)$ we depict in Figure 7.1 the behavior of $g_{\text{ST}}(p, t)$ and $e^{\alpha t} g_{\text{ST}}(p, t)$

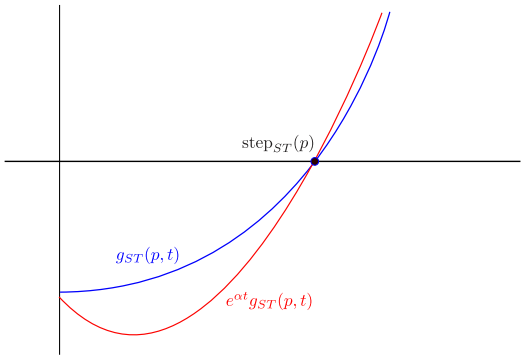

Figure 7.1: Comparison between $g_{\text{ST}}(p,t)$ and $e^{\alpha t} g_{\text{ST}}(p,t)$.

This implies that $\int_0^{\text{step}_{\text{ST}}(p)} e^{\alpha t} g_{\text{ST}}(p,t)dt$ is negative. It is easy to see that $\int_0^t e^{\alpha\tau} g_{\text{ST}}(p,\tau)d\tau$ is an increasing function for $t \geq \text{step}_{\text{ST}}(p)$ taking the derivative. Since $d/dt \int_0^t e^{\alpha\tau} g_{\text{ST}}(p,\tau)d\tau = e^{\alpha t} g_{\text{ST}}(p,t)$ we observe that $\text{step}_{\text{ST}}(p)$ is precisely the critical point of $\int_0^t e^{\alpha\tau} g_{\text{ST}}(p,\tau)d\tau$. Therefore, the solution of the *performance-based* triggering condition has to be bigger than $\text{step}_{\text{ST}}(p)$. In Figure 7.2 we illustrate the behavior of the function $\int_0^t e^{\alpha\tau} g_{\text{ST}}(p,\tau)d\tau$.

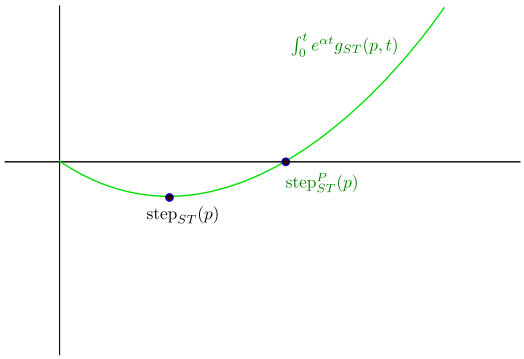

Figure 7.2: Plot of the function $\int_0^t e^{\alpha\tau} g_{\text{ST}}(p,\tau)d\tau$.

□

Remark 7. *Even though we will not prove it here, we will state, for completeness, that it is possible to show that the performance-based trigger is actually strictly better than the performance-based trigger, i.e, there exists* $\epsilon > 0$ *such that*

$$step^{P}_{ST}(p) - step_{ST}(p) \geq \epsilon.$$

Intuitively this makes sense, because the critical point of $\int_0^t e^{\alpha\tau} g_{\mathrm{ST}}(p,\tau)d\tau$ cannot be its zero if the function takes initially negative values.

7.1.1 Simulations

Here we include some simulations that show the improvement of the performance-based algorithm with respect to the derivative-based one. In both cases we assume the objective is quadratic so that we can easily compute L and μ. The MATLAB code used for these simulations is included in Appendix C. For the first case we observe only a slight improvement. In the second case the function is ill-conditioned and the improvement of the *performance-based* trigger with respect to the *derivative-based* one increases over time.

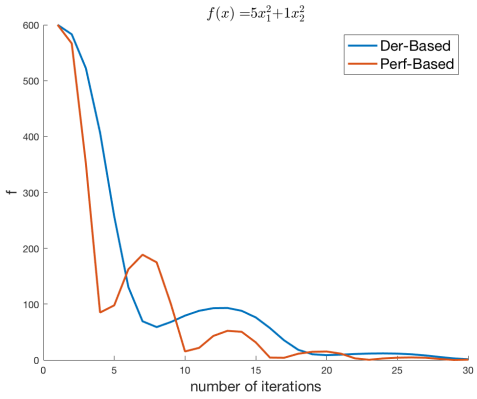

Figure 7.3: Value of $f(x) = 5x^2 + y^2$ in terms of the number of iterations for the *derivative-based* algorithm and the *performance-based* algorithm

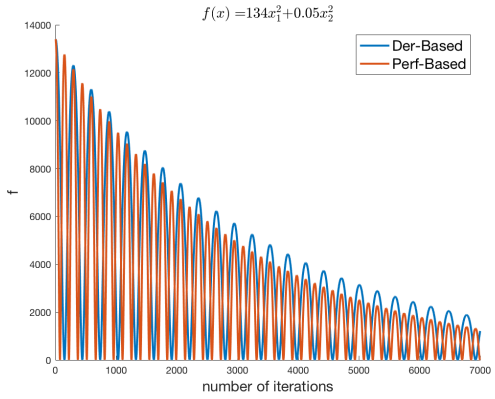

Figure 7.4: Value of $f(x,y) = 134x^2 + 0.05y^2$ in terms of the number of iterations for the *derivative-based* algorithm and the *performance-based* algorithm

7.2 Efficient Use of Sampled-Data Information

In this section we are going to use the information on $\hat{x}$ and $\hat{v}$ to find a better approximation of the gradient term in the *heavy-ball* dynamics.

Let us introduce the following a-dependent family of vector fields:

$$X^a_{\text{hb}}(x,v) = \begin{bmatrix} v \\ -2\sqrt{\mu}v - (1+\sqrt{\mu s})\nabla f(x+av) \end{bmatrix}, \tag{7.5}$$

for $a \geq 0$ where we modified the term $\nabla f(x)$ by with the term $\nabla f(x+av)$.

The following theorem states that the continuous dynamics introduced in 7.5 satisfy the same Lyapunov decay as 6.7 when $a \geq 0$ is small enough. This should come as no surprise, since the solution of a differential equation is continuous with respect to parameters of the vector field. The value of Theorem 26 is that it gives an explicit range of values of a for which this Lyapunov decay occurs.

Theorem 26. *There exists a^* (computed explicitly along the the proof) such that for $0 \leq a \leq a^*$ the solutions of the dynamical system X^a_{hb} satisfy*

$$\dot{V} + \alpha V = \langle \nabla V(p), X^a_{\text{hb}}(p) \rangle + \alpha V(p) \leq 0,$$

for $\alpha \in [0, \frac{\sqrt{\mu}}{4}]$.

Proof. We have

$$\begin{aligned}
\langle \nabla V(p), X^a_{\mathrm{hb}}(p)\rangle + \alpha V(p) &= (1+\sqrt{\mu s})\langle \nabla f(x), v\rangle + \sqrt{\mu}\,\|v\|^2 - 2\sqrt{\mu}\,\|v\|^2 \\
&\quad - (1+\sqrt{\mu s})\langle \nabla f(x+av), v\rangle \\
&\quad - \sqrt{\mu}(1+\sqrt{\mu s})\langle \nabla f(x+av), x - x_*\rangle + \alpha V(x,v) \\
&= (1+\sqrt{\mu s})\langle \nabla f(x), v\rangle - \sqrt{\mu}\,\|v\|^2 + \alpha V(x,v) \\
&\quad - (1+\sqrt{\mu s})\langle \nabla f(x+av) - \nabla f(x) + \nabla f(x), v\rangle \\
&\quad - \sqrt{\mu}(1+\sqrt{\mu s})\langle \nabla f(x+av) - \nabla f(x) + \nabla f(x), x - x_*\rangle \\
&= \underbrace{-\sqrt{\mu}\,\|v\|^2 - \sqrt{\mu}(1+\sqrt{\mu s})\langle \nabla f(x), x - x_*\rangle + \alpha V(p)}_{\text{Derivative-Based Term}} \\
&\quad \underbrace{-(1+\sqrt{\mu s})\langle \nabla f(x+av) - \nabla f(x), v\rangle}_{\text{Term I}} \\
&\quad \underbrace{-\sqrt{\mu}(1+\sqrt{\mu s})\langle \nabla f(x+av) - \nabla f(x), x - x_*\rangle}_{\text{Term II}}
\end{aligned}$$

We observe that "Derivative-Based Term" is the term $\langle \nabla V(p), X_{\mathrm{hb}}\rangle + \alpha V(p)$. Therefore, "Term I" and "Term II" can be seen as perturbations of the original bounds (they vanish for $a = 0$). By strong-convexity of f

$$-\langle \nabla f(x+av) - \nabla f(x), v\rangle \le -a\mu\,\|v\|^2$$

" Term I" is always helping to maintain $\dot V + \alpha V \le 0$. The interpretation of "Term II" is harder. Intuitively, this term would become negative when moving towards the minimizer under reasonable assumptions.
It is easy to see that

$$\begin{aligned}
\text{Derivative-Based Term} &\le (\alpha\frac{3}{4} - \sqrt{\mu})\,\|v\|^2 \\
&\quad + \big((1+\sqrt{\mu s})\frac{\alpha - \sqrt{\mu}}{2L} \\
&\quad + (\alpha 2\mu - \frac{\sqrt{\mu}(1+\sqrt{\mu s})\mu)}{2})\frac{1}{L^2}\big)\,\|\nabla f(x)\|^2\,, \\
\text{Term I} &\le -a(1+\sqrt{\mu s})\mu\,\|v\|^2\,, \\
\text{Term II} &\le \frac{a}{\mu}\sqrt{\mu}(1+\sqrt{\mu s})L\,\|v\|\,\|\nabla f(x)\|\,.
\end{aligned}$$

Therefore,

$$\begin{aligned}\dot{V} + \alpha V \leq (\alpha 2\mu - \frac{\sqrt{\mu}(1+\sqrt{\mu s})\mu)}{2})\frac{1}{L^2}\big) \left\|\nabla f(x)\right\|^2 \\ + (\alpha\frac{3}{4} - \sqrt{\mu}) \left\|v\right\|^2 + \big((1+\sqrt{\mu s})\frac{\alpha - \sqrt{\mu}}{2L} \\ + a\big(- (1+\sqrt{\mu s})\mu \left\|v\right\|^2 \\ + \frac{1}{\mu}\sqrt{\mu}(1+\sqrt{\mu s})L \left\|v\right\| \left\|\nabla f(x)\right\|\big)\big).\end{aligned}$$

Define

$$\begin{aligned}\beta_1 &= (1+\sqrt{\mu s})\mu, \\ \beta_2 &= \frac{1}{\mu}\sqrt{\mu}(1+\sqrt{\mu s})L, \\ \beta_3 &= -(\alpha\frac{3}{4} - \sqrt{\mu}), \\ \beta_4 &= -\big((1+\sqrt{\mu s})\frac{\alpha - \sqrt{\mu}}{2L} + (\alpha 2\mu - \frac{\sqrt{\mu}(1+\sqrt{\mu s})\mu)}{2})\frac{1}{L^2}\big),\end{aligned}$$

and thus

$$\begin{aligned}\dot{V} + \alpha V \leq a\big(- \beta \left\|v\right\|^2 + \beta_2 \left\|v\right\| \left\|\nabla f(x)\right\| \big) \\ - \beta_3 \left\|v\right\|^2 - \beta_4 \left\|\nabla f(x)\right\|^2,\end{aligned}$$

where the β_i's are all positive. So, if we guarantee that

$$a\big(- \beta_1 \left\|v\right\|^2 + \beta_2 \left\|v\right\| \left\|\nabla f(x)\right\| \big) - \beta_3 \left\|v\right\|^2 - \beta_4 \left\|\nabla f(x)\right\|^2 \leq 0$$

then we can ensure the desired Lypaunov decay. Since $a \geq 0$ expression 7.6 is satisfied if $-\beta_1 \left\|v\right\|^2 + \beta_2 \left\|v\right\| \left\|\nabla f(x)\right\| \leq 0$. Thus, we only need to study the complementary case, $-\beta \left\|v\right\|^2 + \beta_2 \left\|v\right\| \left\|\nabla f(x)\right\| > 0$, in which 7.6 can be rewritten as

$$a \leq \frac{\beta_3 \left\|v\right\|^2 + \beta_4 \left\|\nabla f(x)\right\|^2}{-\beta_1 \left\|v\right\|^2 + \beta_2 \left\|v\right\| \left\|\nabla f(x)\right\|}.$$

Finally, we show that $\dfrac{\beta_3 \left\|v\right\|^2 + \beta_4 \left\|\nabla f(x)\right\|^2}{-\beta_1 \left\|v\right\|^2 + \beta_2 \left\|v\right\| \left\|\nabla f(x)\right\|}$ is positively lower bounded. We introduce the following auxiliary expression (where we substituted $\left\|v\right\| = z_1$ and $\left\|\nabla f(x)\right\| = z_2$)

$$\frac{\beta_3 z_1^2 + \beta_4 z_2^2}{-\beta_1 z_1^2 + \beta_2 z_1 z_2}.$$

Dividing by z_1^2 in the numerator and denominator results in

$$\frac{\beta_3 + \beta_4(\frac{z_2}{z_1})^2}{-\beta_1 + \beta_2 \frac{z_2}{z_1}},$$

and renaming $\dfrac{z_2}{z_1} = z$ leads us to the function

$$g(z) = \frac{\beta_3 + \beta_4 z^2}{-\beta_1 + \beta_2 z}.$$

It's easy to see by elementary calculus that this one-variable function is lower bounded. □

Thus, we have proved that the new dynamics 7.5 converges at least at the same rate as the standard high-resolution *heavy-ball* dynamics. It seems convenient therefore to develop triggering algorithms like 3. In order to proceed in that direction we need to upper bound the Lyapunov decay:

Proposition 27 (Triggered-Sampled discretization of 7.5)**.** *For the dynamics 7.5, let* $\dot{p}(t) = X^a_{\text{hb}}(\hat{p})$, $p(0) = \hat{p} = [\hat{x}, \hat{v}]$, *then*

$$\begin{aligned}
&\tfrac{d}{dt}V(p(t)) + \alpha V(p(t)) \\
&= \langle \nabla V(\hat{p} + tX^a_{\text{hb}}(\hat{p})), X^a_{\text{hb}}(\hat{p})\rangle + \alpha V(\hat{p} + tX_{\text{hb}}(\hat{p})) \\
&= \underbrace{\langle \nabla V(\hat{p} + tX^a_{\text{hb}}(\hat{p})) - \nabla V(\hat{p}), X^a_{\text{hb}}(\bar{p})\rangle}_{\text{Term}^{\text{S}}\ \text{I}} \\
&+ \underbrace{\alpha(V(\hat{p} + tX^a_{\text{hb}}(\hat{p})) - V(\hat{p}))}_{\text{Term}^{\text{S}}\ \text{II}} + \underbrace{\langle \nabla V(\hat{p}), X^a_{\text{hb}}(\bar{p})\rangle + \alpha V(\hat{p})}_{\text{Term}^{\text{S}}\ \text{III}}.
\end{aligned}$$

Then, the following bounds hold:

1. $\text{Term}^{\text{S}}\ \text{I} \leq A^S_{\text{ET}}(\hat{p}, a, t) \leq A^S_{\text{ST}}(\hat{p}, a)t$;
2. $\text{Term}^{\text{S}}\ \text{II} \leq BC^S_{\text{ET}}(\hat{p}, a, t) \leq B^S_{\text{ST}}(\hat{p}, a)t + C^S_{\text{ST}}(\hat{p}, a)t^2$;
3. $\text{Term}^{\text{S}}\ \text{III} \leq D^S_{\text{ET}}(\hat{p}, a, t) = D^S_{\text{ST}}(\hat{p}, a)$,

and

$$\begin{aligned}
A^S_{\text{ET}}(\hat{p}, a, t) = (1 + \sqrt{\mu s})&\langle \nabla f(\hat{x} + t\hat{v}) - \nabla f(\hat{x}), \hat{v}\rangle \\
&+ 2\mu t \left\|\hat{v}\right\|^2 + t2\sqrt{\mu}(1 + \sqrt{\mu s})\langle \nabla f(\hat{x} + a\hat{v}), \hat{v}\rangle \\
&+ t(1 + \sqrt{\mu s})^2 \left\|\nabla f(\hat{x} + a\hat{v})\right\|^2
\end{aligned}$$

$$
\begin{aligned}
BC^{S}_{\mathrm{ET}}(\hat{p}, a, t) = (1+\sqrt{\mu s})(f(\hat{x}+t\hat{v}) - f(\hat{x})) + \\
&+ t^2 \frac{1}{4} \left\| -2\sqrt{\mu}\hat{v} - (1+\sqrt{\mu s})\nabla f(\hat{x}+a\hat{v}) \right\|^2 \\
&+ -t(1+\sqrt{\mu s})\langle \hat{v}, \nabla f(\hat{x}+a\hat{v})\rangle - t\sqrt{\mu}\left\|\hat{v}\right\|^2 \\
&+ t^2 \frac{1}{4} \left\| (1+\sqrt{\mu s})\nabla f(\hat{x}+a\hat{v}) \right\|^2 \\
&- t\sqrt{\mu}(1+\sqrt{\mu s})\left\|\nabla f(\hat{x}+a\hat{v})\right\|^2 / L
\end{aligned}
$$

$$
\begin{aligned}
D^{S}_{\mathrm{ET}}(\hat{p}, a) = D^{S}_{\mathrm{ST}} = (\alpha 2\mu - \frac{\sqrt{\mu}(1+\sqrt{\mu s})\mu)}{2}\frac{1}{L^2}\big) \left\|\nabla f(x)\right\|^2 \\
&+ (\alpha\frac{3}{4} - \sqrt{\mu})\left\|v\right\|^2 + \big((1+\sqrt{\mu s})\frac{\alpha - \sqrt{\mu}}{2L} \\
&+ a\big(- (1+\sqrt{\mu s})\mu \left\|v\right\|^2 \\
&+ \frac{1}{\mu}\sqrt{\mu}(1+\sqrt{\mu s})L\left\|v\right\|\left\|\nabla f(x)\right\|\big).
\end{aligned}
$$

$$
\begin{aligned}
A^{S}_{\mathrm{ST}}(\hat{p}, a) = (1+\sqrt{\mu s})L\left\|v\right\|^2 + 2\mu\left\|\hat{v}\right\|^2 \\
&+ 2\sqrt{\mu}(1+\sqrt{\mu s})\langle \nabla f(\hat{x}+a\hat{v}), \hat{v}\rangle \\
&+ (1+\sqrt{\mu s})^2\left\|\nabla f(\hat{x}+a\hat{v})\right\|^2
\end{aligned}
$$

$$
\begin{aligned}
B^{S}_{\mathrm{ST}}(\hat{p}, a) = -\sqrt{\mu}\left\|\hat{v}\right\|^2 \\
&- \sqrt{\mu}(1+\sqrt{\mu s})\frac{1}{L}\left\|\nabla f(\hat{x}+a\hat{v})\right\|^2,
\end{aligned}
$$

$$
\begin{aligned}
C^{S}_{\mathrm{ST}}(\hat{p}, a) = (1+\sqrt{\mu s})\frac{L}{2}\left\|\hat{v}\right\|^2 \\
&+ \frac{1}{4}\left\| -2\sqrt{\mu}\hat{v} - (1+\sqrt{\mu s})\nabla f(\hat{x}+a\hat{v})\right\|^2 \\
&+ \frac{1}{4}\left\| -(1+\sqrt{\mu s})\nabla f(\hat{x}+a\hat{v})\right\|^2,
\end{aligned}
$$

Proof. The computations are included in Appendix B. □

Next, we can define:

$$g^S_{\mathrm{ET}}(\hat{p}, a, t) = A^S_{\mathrm{ET}}(\hat{p}, a, t) + BC^S_{\mathrm{ET}}(\hat{p}, a, t) + D^S_{\mathrm{ET}}(\hat{p}, a, t),$$
$$g^S_{\mathrm{ST}}(\hat{p}, a, t) = C^S_{\mathrm{ST}}(\hat{p}, a)t^2 + (A^S_{\mathrm{ST}}(\hat{p}, a) + B^S_{\mathrm{ST}}(\hat{p}, a))t + D^S_{\mathrm{ST}}(\hat{p}, a),$$

and we have the bounds

$$\frac{d}{dt}V(p(t)) + \alpha V(p(t)) \leq g^S_{\mathrm{ET}}(\hat{p}, t) \leq g^S_{\mathrm{ST}}(\hat{p}, a, t).$$

Using the bounds computed in Proposition 27 we can determine the stepsize starting from $\hat{p}$. We set:

$$\mathrm{step}^S_{\#}(\hat{p}, a) = \min\{t \mid t > 0 \text{ such that } g^S_{\#}(\hat{p}, a, t) = 0\} \tag{7.6}$$

and implement Algorithm 5, whose properties are gathered in proposition 28.

Algorithm 5: Predictive, Forward-Euler

Initialization: Initial point (p_0), convergence rate (α), objective function (f), tolerance (ϵ), $a \leq a^*$;
Set: $k = 0$;
while $\|\nabla f(x_k)\| \geq \epsilon$ **do**
  Compute stepsize t_k according to (7.6);
  Compute next iterate $p_{k+1} = p_k + t_k X^a_{\mathrm{hb}}(p_k)$;
  Set $k = k + 1$
end

Proposition 28 (Non-Zeno and Convergence-Rate for algorithm 5)**.** *Algorithm 5 satisfies*

1. *The stepsize is uniformly lower bounded*
2. *The algorithm satisfies the convergence rate*

$$f(x_{k+1}) - f(x_*) = \mathcal{O}(e^{-\alpha \sum_{i=0}^{k}})$$

Proof. The proof follows follows a pattern similar to the one done in chapter 6 for the $a = 0$ case. The key point is showing that $D^S_{\mathrm{ET}} \leq 0$, which follows from Theorem 26. □

7.2.1 Simulations

Even though we haven't theoretically proved that Algorithm 5 with $a > 0$ has better convergence properties than the same algorithm with $a = 0$ (i,e, Algorithm 3), here we present some simulations that suggest that this is the case. Once again we consider only quadratic objectives so that we are able to compute L and μ easily. The MATLAB code used for these simulations is included in Appendix C.
The first one is for the well conditioned $f(x, y) = 5x^2 + y^2$. The second simulation is for the ill-conditioned $f(x, y) = 134x^2 + 0.05y^2$. In this case, the improvement seems to be even bigger than the improvement we got from using Algorithm 4.

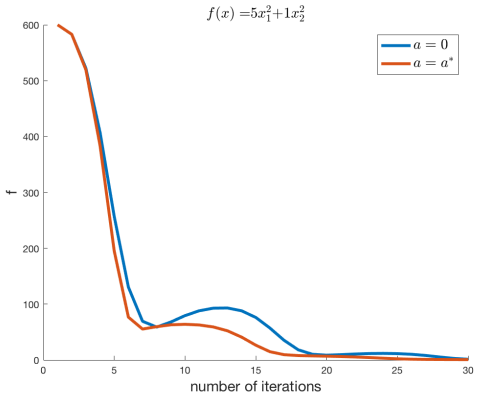

Figure 7.5: Value of $f(x, y) = 5x^2 + y^2$ in terms of the number of iterations for algorithm 5 with $a = a^*$

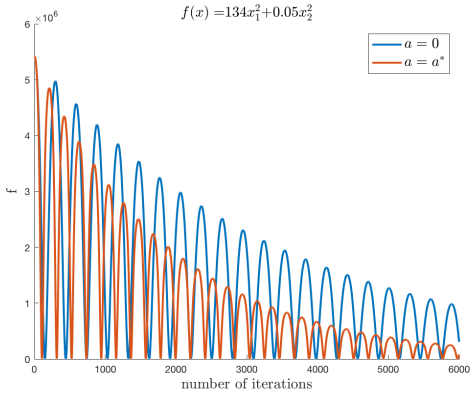

Figure 7.6: Value of $f(x, y) = 134x^2 + 0.05y^2$ in terms of the number of iterations for Algorithm 5 with $a = a^*$ and Algorithm 3

Even though most of the simulations we have tried show that algorithm 5 doesn't exhibit acceleration, an interesting phenomenon occurs when we consider values of a outside the range where we have proved convergence. Figure 7.7 shows a simple example where considering $a = 5a^*$, algorithm 5 performs significantly better than with $a = a^*$.
Note also another interesting phenomenon from Figure 7.7: the *overshoot* of Polyak's Heavy-Ball is significantly bigger than that of Algorithm 5.

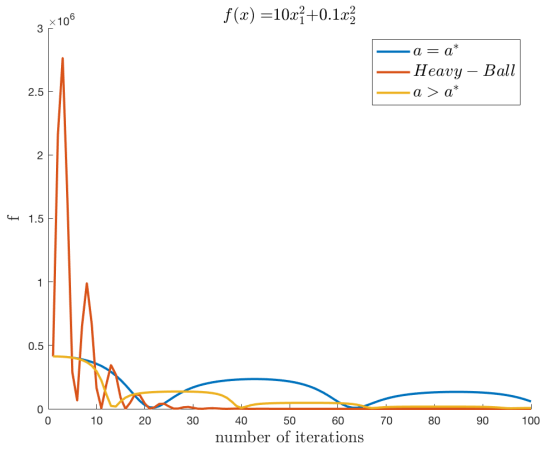

Figure 7.7: Value of $f(x, y) = 100x^2 + y^2$ in terms of the number of iterations for algorithm 5 and Polyak's Heavy Ball

7.3 Adaptive Sampling

In the previous section we designed new dynamics that use a uniform value of a. Here we introduce an adaptive version where a adjusts to the current state of the algorithm. The motivation behind using an adaptive a comes from the fact that the value of a^* from 26 has been calculated by finding the global minimum of the function

$$\frac{\beta_3 \|v\|^2 + \beta_4 \|\nabla f(x)\|^2}{-\beta_1 \|v\|^2 + \beta_2 \|v\| \|\nabla f(x)\|}$$

However, if we restrict this function on a specific subregion of the state space, this minimal value can be improved. This is generally a good thing, since many simulations, like 7.7 suggest that running algorithm 4 with values of a greater than a^* leads to improved convergence rate. This should come as no surprise, since 28 only specifies a range for the values of a for which the algorithm is convergent, but this range might not be tight. It would be interesting to try to find better theoretical upper bounds but to the best of our knowledge the bound provided in B (which is an improved version on the one from 26) is the best one available.

Therefore, we will establish a way to use an adaptive value of a that depends on the region of the state space of the current iterate. We motivate this from the continuous framework, computing conditions on (x, v) such that $\langle \nabla V(p(t)), X^a_{\text{hb}}(p(t)) \rangle + \alpha V(p(t)) \leq 0$ is satisfied in those regions. We choose two sequences $\{n_k\}$ and $\{m_k\}$ which are positive, monotonically increasing and satisfy

$$\lim_{k\to\infty} n_k = \lim_{k\to\infty} m_k = \infty.$$

Then we split the state space into regions R_{ij} defined by

$$\begin{aligned} R_{ij} = \{&(x, v) \text{ such that } n_i \leq \|\nabla f(x)\|^2 < n_{i+1}, \\ &m_j \leq \|v\|^2 < m_{j+1}\}. \end{aligned}$$

Observe that $R_{ij} \cap R_{kl} = \varnothing$ and $\cup_{ij} R_{ij} = \mathbb{R}^{2n}$. Next, we show how to compute a value of a for any of these regions satisfying the Lyapunov decay $\dot{V} + \alpha V \leq 0$. For the sake of simplicity fix i and j, and let us denote

$$\begin{aligned} \nabla f_{\min} &= n_i, \\ \nabla f_{\max} &= n_{i+1}, \\ v_{\min} &= m_i, \\ v_{\max} &= m_{i+1} \end{aligned}$$

and so we focus on the set of points (x, v) satisfying

$$v_{\min} \leq \|v\| < v_{\max}, \quad \nabla f_{\min} \leq \|\nabla f(x)\| < \nabla f_{\max}. \tag{7.7}$$

Following bounds similar to the previous ones, we can upper bound 7.6 by

$$a(-\beta_1 v_{\min}^2 + \beta_2 v_{\max} \nabla f_{\max}) - \beta_3 v_{\min}^2 - \beta_4 \nabla f_{\min}^2.$$

If $a(-\beta_1 v_{\min}^2 + \beta_2 v_{\max} \nabla f_{\max}) \leq 0$ then any value of a satisfies the decay condition $\dot{V} + \alpha V$ on the region specified by 7.7. If $a(-\beta_1 v_{\min}^2 + \beta_2 v_{\max} \nabla f_{\max}) \geq 0$ then any value of a satisfying

$$a \leq \frac{\beta_3 v_{\min}^2 + \beta_4 \nabla f_{\min}^2}{-\beta_1 v_{\min} + \beta_2 v_{\max} \nabla f_{\max}}$$

implies the decay condition $\dot{V} + \alpha V$ on the region specified by 7.7. Therefore, we can define the switched or hybrid system given by

$$\begin{bmatrix} \dot{x} \\ \dot{v} \end{bmatrix} = X^a_{\text{hb}}(x, v) \quad \text{where } a = \frac{\beta_3 v_{\min}^2 + \beta_4 \nabla f_{\min}^2}{-\beta_1 v_{\min} + \beta_2 v_{\max} \nabla f_{\max}} \tag{7.8}$$

and

$$\nabla f_{\min} = n_i \quad i = \arg\max_i n_i \text{ such that } n_i \leq \|\nabla f(x)\|^2, \tag{7.9}$$

$$\nabla f_{\max} = n_{i+1}, \tag{7.10}$$

$$v_{\min} = n_j \quad j = \arg\max_j n_i \text{ such that } m_i \leq \|v\|, \tag{7.11}$$

$$v_{\max} = n_{j+1}. \tag{7.12}$$

By design the proposed dynamics satisfy the following proposition.

Proposition 29 (Convergence of Triggered-Sampled Dynamics 7.8-7.12)**.** *The dynamical system described in 7.8-7.12 converges to the minimizer, x_*, satisfying the Lyapunov decay condition*

$$\dot{V} + \alpha V \leq 0$$

Proof. By design the dynamics 7.8-7.12 satisfy the Lyapunov decay condition in any regions, so the result follows. □

This proposition suggests that we can design an algorithm that uses an adaptive a by taking into account the current state and still guarantee Zeno behavior.

Informal Description

> At each iterate, Algorithm 6 checks whether $D_{ET}(\hat{p}, a) < 0$ and the stepsize is greater than a pre-specified lower bound. If this is the case, the value of a is augmented by a certain factor η_1 (inspired by the heuristic outlined above that a bigger value of a might lead to an improved convergence rate). If one of the two conditions is not fulfilled, the value of a is decreased by a certain factor η_2, arguing that eventually, if a gets close enough to zero, the two conditions will be satisfied. The value of a is reused throughout the iterations so that the algorithm has *memory* of the values of a used in previous iterations.

Algorithm 6: Adaptive Sampling Algorithm

Initialization: Initial point (p_0), convergence rate (α), objective function (f), tolerance (ϵ), increase rate ($\eta_1 > 1$), decrease rate ($\eta_2 < 1$), lower bound of the stepsize ($\hat{t}$), initial a (a_0);
Set: $k = 0$;
while $\|\nabla f(x_k)\| \geq \epsilon$ **do**
  Compute stepsize t_k according to 7.4;
  if $D^a_{\mathrm{ET}}(\hat{p}, a, t) < 0$ *and* $\hat{t} \leq t_k$ **then**
    Compute next iterate $p_{k+1} = p_k + t_k X_{\mathrm{hb}}(p_k)$;
    $a = a\beta_1$;
  else
    while $D^a_{\mathrm{ET}}(\hat{p}, a, t) \geq 0$ *or* $\hat{t} \geq t_k$ **do**
      $a = a\beta_2$;
    end
    Compute stepsize t_k according to (7.4);
    Compute next iterate $p_{k+1} = p_k + t_k X_{\mathrm{hb}}(p_k)$;
  Set $k = k + 1$
end

The properties of Algorithm 6 are gathered in the following proposition:

Proposition 30 (Non-Zeno and Convergence-Rate for Algorithm 6)**.** *Algorithm 6 satisfies*

1. *It is executable*
2. *The stepsize is uniformly lower bounded*
3. *The algorithm satisfies the convergence rate*

$$f(x_{k+1}) - f(x_*) = \mathcal{O}(e^{-\alpha \sum_{i=0}^{k}})$$

Proof. Observe that Algorithm 6 has the following properties:

- At any step, as long as $D_{ET} < 0$ a new stepsize t_k can be computed, although we don't know if this step is lower-bounded for $a > a^*$.

- If the lower bound of the stepsize $\hat{t}$ is less than the bound provided in Proposition 28, then the algorithm will satisfy the condition $t_k > \hat{t}$

if a is decreased enough (by continuity). This rules out Zeno behaviour.

- For a sufficiently small we are in the conditions of Proposition 28, which guarantee that the algorithm always terminates, i.e, for small enough a, $D_{ET} < 0$ and $t_k > \hat{t}$ will be satisfied.
  The proof follows from these observations.

□

7.3.1 Simulations

Here we present two simulations that show that an algorithm with adaptive a has better convergence properties than an algorithm with constant $a > 0$. The MATLAB code used for these simulations is included in appendix C.

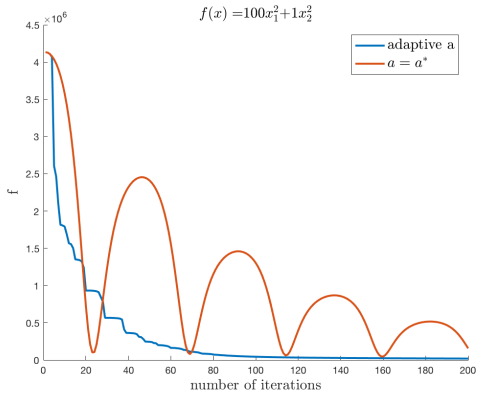

Figure 7.8: Value of $f(x, y) = 100x^2 + y^2$ in terms of the number of iterations for algorithm 6 (adaptive a) and algorithm 5(constant $a > 0$), $\eta_1 = 5$, $\eta_2 = 0.5$, $x_0 = [201, 304]$, $v_0 = -\frac{2\sqrt{s}\nabla f(x_0)}{1+\sqrt{\mu s}}$

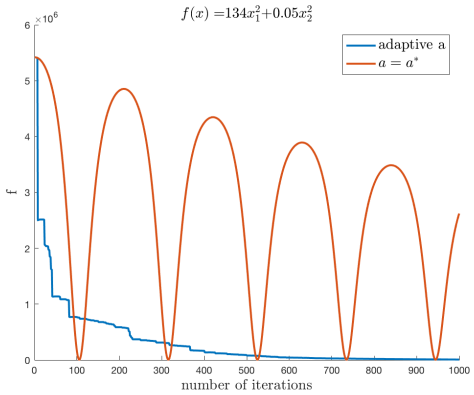

Figure 7.9: Value of $f(x, y) = 134x^2 + 0.05y^2$ in terms of the number of iterations for algorithm 6(adaptive a) and algorithm 5 (constant $a = a^* > 0$, $\eta_1 = 5$, $\eta_2 = 0.5$), $x_0 = [201, 304]$, $v_0 = -\frac{2\sqrt{s}\nabla f(x_0)}{1+\sqrt{\mu s}}$

The improvement in both cases is huge. Note that the characteristic oscillations of the heavy-ball dynamics are avoided.
It should be noted though that we lack a principled way to select the hyperparameters η_1 and η_2 in order to ensure the maximal improvement on the

constant a case. For the simulations above, they have been selected by *trial and error*.

7.4 High-Order Integrators

In this section we will derive a triggered implemetation for a more accurate integrator than the *sample-and-hold* (or *zero-order-hold*) one we have been using up to this point.

$$\begin{aligned} X_{\text{hb}}(x,v) &= \begin{bmatrix} v \\ -2\sqrt{\mu}v - (1+\sqrt{\mu s})\nabla f(x)) \end{bmatrix} \\ &= \underbrace{\begin{bmatrix} v \\ -2\sqrt{\mu}v \end{bmatrix}}_{\text{Linear Term}} + \underbrace{\begin{bmatrix} 0 \\ -(1+\sqrt{\mu s})\nabla f(x)) \end{bmatrix}}_{\text{Non-linear Term}}. \end{aligned} \tag{7.13}$$

Since the non-linear part is the one complicating the dynamics, *i.e.* hindering the closed-form solution of the dynamics, it is appealing to design an integrator which takes this part as a constant. That is, given $\hat{x}$ we introduce the family of dynamical systems

$$X_{\text{hb}}^{H,\hat{x}}(x,v) = \begin{bmatrix} v \\ -2\sqrt{\mu}v - (1+\sqrt{\mu s})\nabla f(\hat{x}) \end{bmatrix} \tag{7.14}$$

denoted by *first-order-hold dynamics*. These dynamical systems are in-homogeneous linear dynamical systems and therefore integrable by the method of variation of constants. More precisely, the flow with initial condition $\hat{p} = [\hat{x}, \hat{v}]$ is given by

$$x(t,\hat{x},\hat{v}) = K_1(\hat{x},\hat{v})e^{-2\sqrt{\mu}t} - \frac{(1+\sqrt{\mu s})\nabla f(\hat{x})}{2\sqrt{\mu}}t + K_2(\hat{x},\hat{v}) \tag{7.15}$$

$$v(t,\hat{x},\hat{v}) = -2\sqrt{\mu}K_1(\hat{x},\hat{v})e^{-2\sqrt{\mu}t} - \frac{1+\sqrt{\mu s}}{2\sqrt{\mu}}\nabla f(\hat{x}) \tag{7.16}$$

where

$$\begin{aligned} K_1(\hat{x},\hat{v}) &= -\frac{\hat{v}}{2\sqrt{\mu}} - \frac{1+\sqrt{\mu s}}{4\mu}\nabla f(\hat{x}) \\ K_2(\hat{x},\hat{v}) &= \hat{x} + \frac{\hat{v}}{2\sqrt{\mu}} + \frac{1+\sqrt{\mu s}}{4\mu}\nabla f(\hat{x}) \end{aligned}$$

This is what we will call the *first-order-hold* integrator, or *high-order-hold* integrator. In order to develop a discretization of the continuous heavy-ball dynamics where we use the dynamics 7.15-7.16 as an integrator, we need to compute the evolution of $\dot{V} + \alpha V$ along the proposed dynamics. We proceed only in the ET-case for the sake of simplicity, but analogous results can be computed for the ST-case.

Proposition 31 (Event/Self-triggered-HOH discretization of heavy-ball dynamics)**.** *For the predicted high-order-hold dynamics* $\dot{p} = X_{\text{hb}}^{H,\hat{x}}(p(t))$, $p(0) = \hat{p} = [\hat{x}, \hat{v}]$ *and* $0 \leq \alpha \leq \sqrt{\mu}/4$, *let*

$$\begin{aligned}
&\tfrac{d}{dt}V(p(t)) + \alpha V(p(t)) \\
&= \langle \nabla V(p(t)), X_{\text{hb}}^{H,\hat{x}}(p(t))\rangle + \alpha V(p(t)) \\
&= \underbrace{\langle \nabla V(p(t)) - \nabla V(\hat{p}), X_{\text{hb}}^{H,\hat{x}}(p(t))\rangle}_{\text{Term}^{\text{H}}\ \text{I}} \\
&+ \underbrace{\langle \nabla V(p(0)), X_{\text{hb}}^{H,\hat{x}}(p(t)) - X_{\text{hb}}^{HOH,\hat{x}}(\hat{p})\rangle}_{\text{Term}^{\text{H}}\ \text{II}} \\
&+ \underbrace{\alpha(V(p(t)) - V(p(0))}_{\text{Term}^{\text{H}}\ \text{III}} \\
&+ \underbrace{\langle \nabla V(\hat{p}), X_{\text{hb}}^{H,\hat{x}}(\hat{p})\rangle + \alpha V(\hat{p})}_{\text{Term}^{\text{H}}\ \text{IV}}
\end{aligned} \tag{7.17}$$

Then, the following bounds hold:

$$\begin{aligned}
\text{Term}^{\text{H}}\ \text{I} = A_{\text{ET}}^{H} \leq\ & (1 + \sqrt{\mu s})\langle \nabla f(x(t)) - \nabla f(x(0)), v(t)\rangle \\
& - \sqrt{\mu}\langle v(t) - v(0), v(t)\rangle \\
& - (1 + \sqrt{\mu s}\langle v(t) - v(0), \nabla f(\hat{x})\rangle \\
& - \sqrt{\mu}(1 + \sqrt{\mu s})\langle x(t) - x(0), \nabla f(\hat{x})\rangle
\end{aligned}$$

$$\begin{aligned}
\text{Term}^{\text{H}}\ \text{II} = B_{\text{ET}} \leq\ & (1 + \sqrt{\mu s})\langle \nabla f(x), v(t) - v(0)\rangle \\
& - \sqrt{\mu}\langle v(0), v(t) - v(0)\rangle
\end{aligned}$$

$$\begin{aligned}\text{Term}^{\text{H}}\ \text{III} = C_{\text{ET}}^{H} \leq \alpha(1+\sqrt{\mu s})(f(x(t)) - f(\hat{x})) \\ + \frac{\alpha}{4}(\|v(t)\|^2 - \|\hat{v}\|^2) \\ \frac{\alpha}{4}\left\|v(t) - \hat{v} + 2\sqrt{\mu}(x(t)-\hat{x})\right\|^2 \\ + \frac{\alpha}{2}\langle v(t) - \hat{v} + 2\sqrt{\mu}(x(t)-\hat{x}), \hat{v}\rangle + \\ t(1+\sqrt{\mu s})\frac{\left\|\nabla f(\hat{x})\right\|^2}{2\mu}\end{aligned}$$

$$\begin{aligned}\text{Term}^{\text{H}}\ \text{IV} = D_{\text{ET}}^{H} \leq (\alpha\frac{3}{4} - \sqrt{\mu})\left\|\hat{v}\right\|^2 \big((1+\sqrt{\mu s})\frac{\alpha-\sqrt{\mu}}{2L} \\ + (\alpha 2\mu - \frac{\sqrt{\mu}(1+\sqrt{\mu s})\mu)}{2})\frac{1}{L^2}\big)\left\|\nabla f(\hat{x})\right\|^2,\end{aligned}$$

Proof. The proof is presented in Appendix B. □

Using these computations, we define the upper bound, $g_{\text{ET}}(\hat{p}, t)$ in the obvious way:

$$g_{\text{ET}}^{H}(\hat{p}, t) = A_{\text{ET}}^{H}(\hat{p}, t) + B_{\text{ET}}^{H}(\hat{p}, t) + C_{\text{ET}}^{H}(\hat{p}, t) + D_{\text{ET}}^{H}(\hat{p}, t) \tag{7.18}$$

$$\frac{d}{dt}V(p(t)) + \alpha V(p(t)) \leq g_{\text{ET}}(\hat{p}, t). \tag{7.19}$$

This is all we need to determine the stepsize starting from $\hat{p}$. We set:

$$\text{step}_{ET}^{H}(\hat{p}) = \min\{t \mid t > 0 \text{ such that } g_{ET}^{H}(\hat{p}, t) = 0\}$$

and propose the following algorithm:

Theorem 32. *For $0 \leq \alpha \leq \sqrt{\mu}/4$, Algorithm 7 is a variable stepsize integrator with the following properties*

1. *The stepsize is uniformly lower bounded by a positive constant;*
2. *The algorithm satisfies the convergence rate*

$$f(x_{k+1}) - f(x_*) = \mathcal{O}(e^{-\alpha\sum_{i=0}^{k}})$$

Algorithm 7: High-Order-Hold Algorithm

Initialization: Initial point (p_0), convergence rate (α), objective function (f), tolerance (ϵ) ;
Set: $k = 0$;
while $\|\nabla f(x)\| \geq \epsilon$ **do**
  Compute stepsize t_k as $g^H_{ET}(\hat{p}, t) = 0$;
  Compute next iterate according to 7.15 and 7.16 ;
  Set $k = k + 1$
end

Proof. We just sketch the proof. Using

$$\begin{aligned} x(t) - x(0) &= \frac{(1+\sqrt{\mu s})\nabla f(\hat{x})(1-e^{-2t\sqrt{\mu}})}{4\mu} \\ &\quad + \frac{2\hat{v}\sqrt{\mu}(1-e^{-2t\sqrt{\mu}}) - 2(1+\sqrt{\mu s})\nabla f(\hat{x})t\sqrt{\mu}}{4\mu} \\ v(t) - v(0) &= 1/2(e^{-2t\sqrt{\mu}} - 1)(2v(0) + \frac{1+\sqrt{\mu s}}{\sqrt{\mu}}\nabla f(x)) \end{aligned}$$

we may bound $\text{step}^H_{\text{ET}}$ by an expression of the form

$$\text{step}^H_{\text{ET}} \leq (\beta_1 \|\hat{v}\|^2 + \beta_2 \|\nabla f(\hat{x})\|^2)t + D_{ST}(\hat{p})$$

where β_i are strictly positive. To ensure

$$\text{steps}^H_{\text{ET}} \leq 0$$

we just need

$$\begin{aligned} (\beta_1 \|\hat{v}\|^2 + \beta_2 \|\nabla f(\hat{x})\|^2)t + D_{ST}(\hat{p}) &\leq 0 \\ \Leftrightarrow t &\leq \frac{-D_{\text{ST}}}{(\beta_1 \|\hat{v}\|^2 + \beta_2 \|\nabla f(\hat{x})\|^2}. \end{aligned}$$

Finally, using Lemma 1 in the supplementary material of [16] we can proof the result. □

7.4.1 Simulations

First we show how the trajectory followed by the first order hold 7.13 differs from that followed by the zero-order-hold (which is obviously a straight line) and the exact dynamics 6.7:

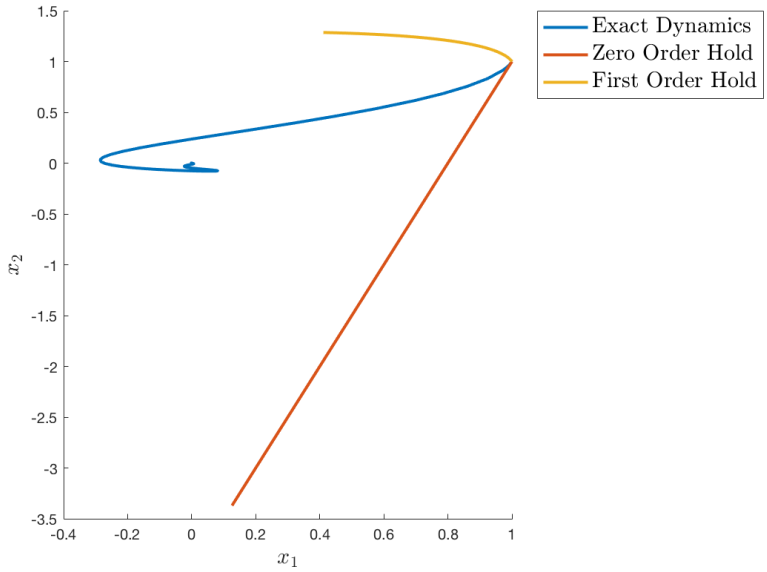

Figure 7.10: Trajectories of Zero-Order-Hold, First-Order-Hold 7.13 and Exact Dynamics for $f(x, y) = 5x^2 + y^2$ for a single iteration with initial conditions $x_0 = [1; 1]$ $v_0 = -0.43[1; 5]$

Even though the trajectory followed by 7.13 is still far from the actual dynamics, it appears to be better than a mere straight line approximation. It should also be noted that as we get closer to the optimum, $\nabla f(x(t))$ gets closer to $\nabla f(\hat{x})$ and 7.13 becomes an increasingly more accurate surrogate for 6.7.

Once again, even though we haven't provided any theoretical results establishing the superiority of Algorithm 7 (first-order-hold, or FOH for short) to algorithm 3 (zero-order-hold, or ZOH for short), we will present some simulations that suggest that this is the case. The MATLAB code used for these simulations is included in appendix C.

The next figure shows the comparison between the *zero-order-hold*-based al-

gorithm and the *first-order-hold*-based one, both for a fairly well conditioned function and an ill-conditioned one.

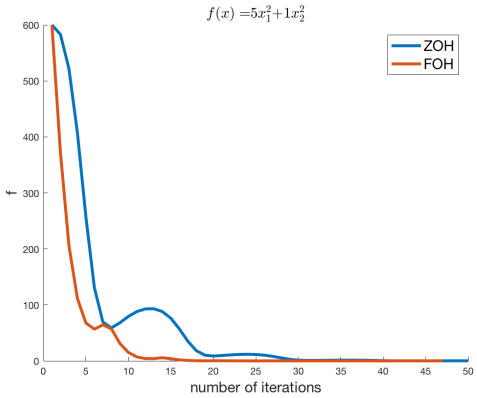

Figure 7.11: Value of $f(x, y) = 5x^2 + y^2$ in terms of the number of iterations for Algorithm 7 (FOH) and algorithm 3(ZOH)

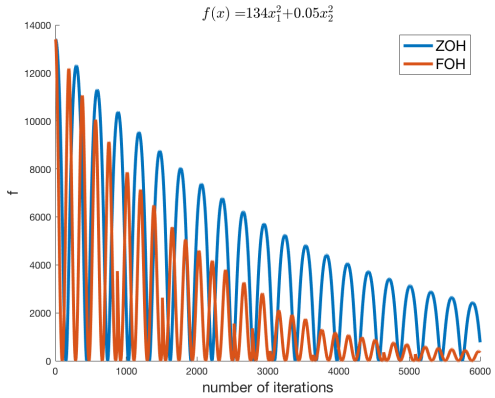

Figure 7.12: Value of $f(x, y) = 134x^2 + 0.05y^2$ in terms of the number of iterations for Algorithm 7(FOH) and Algorithm 3(ZOH)

7.5 Performance-Based Sampled Integrator

Here we will show how to combine approaches 7.1 and 7.2 to obtain a more powerful integrator. It should be noted that the *High-Order-Hold* could also be coupled to it, but we decide not to include it to simplify the computations. The critical point is showing that we can also find an explicit range of values of a for which we can guarantee the desired convergence rate. We will prove that the same range $[0, a^*]$ can be used in the *performance-based* case.
By an analogous argument to the one done in 7.1, the next stepsize can be found by solving the equation:

$$t_{i+1} = \min\{t \mid t > t_i \text{ such that } \int_0^t e^{\alpha t} g_{\#}(p_i, a, t) = 0\}. \tag{7.20}$$

where $g_{\mathrm{ET}}(p, a, t)$ and $g_{\mathrm{ST}}(p, a, t)$ are the ones defined in 7.2. The only condition we need for t_{i+1} to be well defined is $g_{\mathrm{ST}}(p_i, a, 0) = D_{\mathrm{ST}}(p_i, a) < 0$. We can guarantee that by taking $a < a^*$ as we know from 26. We can then

implement the following algorithm:

Algorithm 8: Predictive, Forward-Euler

Initialization: Initial point (p_0), convergence rate (α), objective function (f), tolerance (ϵ), $a \leq a^*$;
Set: $k = 0$;
while $\|\nabla f(x_k)\| \geq \epsilon$ **do**
    Compute stepsize t_k according to (7.20);
    Compute next iterate $p_{k+1} = p_k + t_k X^a_{\text{hb}}(p_k)$;
    Set $k = k + 1$
end

Proposition 33. *(Non-Zeno and Convergence-Rate for 8) Algorithm 8 satisfies:*

1. *The stepsize is uniformly lower bounded*
2. *The algorithm satisfies the convergence rate*

$$f(x_{k+1}) - f(x_*) = \mathcal{O}(e^{-\alpha \sum_{i=0}^{k}})$$

Proof. Define $F(t) = \int_0^t e^{\alpha\tau} g_{\text{ST}}(p, a, \tau) d\tau$.
The first part follows from observing that

$$\frac{d}{dt} F(t) = e^{\alpha t} g_{\text{ST}}(p, a, t)$$

so the critical points of $F(t)$ are the zeros of $g_{\text{ST}}(p, a, t)$ and since $F(t)$ is negative in a neighbourhood of the origin, the first zero of $F(t)$ is greater than its first critical point. Now since we know that the zeros of $g_{\text{ST}}(p, a, t)$ (which are of course $\text{step}^S_{\text{ST}}(p, a)$) are lower bounded, we are done. The second point follows by construction. □

7.5.1 Simulations

Here we show how Algorithm 8 performs better than Algorithm 4 and Algorithm 5 separately in a simple simulation.

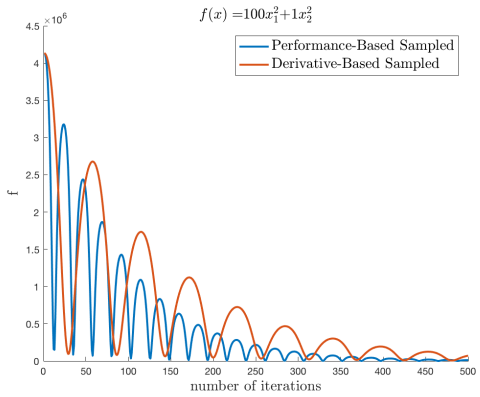

Figure 7.13: Value of $f(x, y) = 100x^2 + y^2$ in terms of the number of iterations for Algorithm 8 and algorithm 5

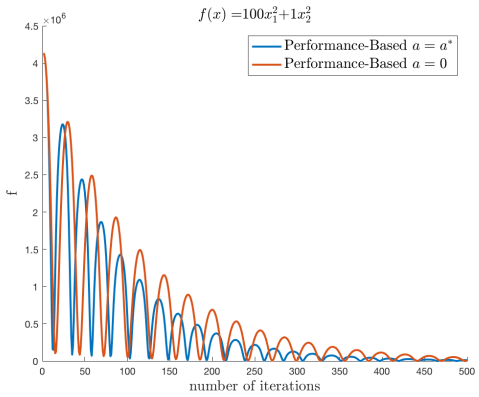

Figure 7.14: Value of $f(x, y) = 100x^2 + y^2$ in terms of the number of iterations for Algorithm 8 and Algorithm 5

Chapter 8

Estimating Sequences from a Continuous-time perspective

This chapter will review the *estimating sequences* technique introduced in chapter 2, but this time we will do so from a continuous setting. Once again, as a recurring theme in this dissertation, we hope that the continuous-time perspective will help us in the analysis.
Most of the ideas covered in this chapter are based on some notes Miguel Vaquero gave me during our discussions at UCSD. I have completed some of the proofs and given coherence to the notes with respect to the rest of the thesis, but the fundamental ideas are his. A similar analysis can be found at [2].
We will start by requiring a certain rate of convergence of the objective function to its optimal value, and we will end up deriving dynamics that ensure such a convergence rate.
Let $f \in \mathscr{S}^2_{\mu,L}$ and as usual we want to solve the unconstrained minimization problem:

$$\min_{x\in\mathbb{R}^n} f(x)$$

We want to construct continuous and discrete dynamics to solve this problem with convergence rate $\mathcal{O}(\lambda(t))$. For convenience, define $\alpha(t)$ such that $\alpha(t) = \frac{1}{\lambda(t)}$ and satisfying $\alpha(0) = 1$, $\dot{\alpha}(t) > 0$ and $\lim_{t\to\infty} \alpha(t) = \infty$.
Let's define the continuous-time equivalent of an estimating sequence:

Definition 34. *A sequence of functions $\{\phi_t(x)\}$ is an estimating sequence if*

$$\phi_t(x) \leq (1 - \frac{1}{\alpha(t)})f(x) + \frac{1}{\alpha(t)}\phi_0(x) \tag{8.1}$$

Remark 8. *$\phi_t(x)$ is an homotopy between $f(x)$ and $\phi_0(x)$*

Proposition 35. *If $x(t)$ is such that*

$$f(x(t)) \leq \min_x \phi_t(x)$$

then

$$f(x(t)) - f(x_*) = \mathcal{O}(\frac{1}{\alpha(t)})$$

Proof.

$$f(x(t)) \leq \min_x \phi_t(x) \leq (1 - \lambda(t))f(x_*) + \lambda(t)\phi_0(x_*)$$
$$\Rightarrow f(x(t)) - f(x_*) \leq \frac{\phi_0(x_*) - f(x_*)}{\alpha(t)} = \mathcal{O}(\frac{1}{\alpha(t)})$$

□

Just like in the discrete case, the obvious question at this point is how to construct these estimating sequences. Drawing inspiration from discrete-time formulas for constructing estimating sequences, the following proposition gives a way to construct them:

Proposition 36. *Let $y(t)$ be an arbitrary curve on $\mathbb{R}^n$. Define:*

$$\phi_t(x) = \frac{\int_0^t f(y(\tau))\dot{\alpha}(\tau)d\tau + \int_0^t \langle \nabla f(y(\tau)), x - y(\tau)\rangle \dot{\alpha}(\tau)d\tau + \phi_0(x)}{\alpha(t)} \tag{8.2}$$

Then, $\phi_t(x)$ is an estimate sequence.

Proof. Recall that if f is convex, then $f(x) \geq f(y) + \nabla f(y)^T(x - y)$. Now since $\dot{\alpha}(t) > 0$ and by monotonicity of the integral:

$$\int_0^t f(y(\tau))\dot{\alpha}(\tau)d\tau + \int_0^t \langle \nabla f(y(\tau)), x - y(\tau)\rangle \dot{\alpha}(\tau)d\tau \leq \int_0^t f(x)\dot{\alpha}(t)d\tau$$
$$= f(x)\Big(\alpha(t) - 1\Big)$$

where in the evaluation of the integral we have used the fact that $\alpha(0) = 1$. Now, substituting in 8.2, we get:

$$\phi_t(x) \leq f(x)(1 - \frac{1}{\alpha(t)}) + \phi_0(x)\frac{1}{\alpha(t)}$$

And therefore $\phi_t(x)$ is an estimating sequence.

□

Now we want to construct a curve $x(t)$ such that $f(x(t)) \leq \min_x \phi_t(x)$ is satisfied. Start by defining:

$$z(t) = \arg\min_x \phi_t(x)$$

Equivalently:

$$z(t) = \arg\min_x \int_0^t \langle \nabla f(y(\tau)), x - y(\tau)\rangle \dot{\alpha}(\tau) d\tau + \phi_0(x)$$

Let's find out the dynamics of $z(t)$:

$$\begin{aligned} \frac{d}{dx}\Big(\int_0^t \langle \nabla f(y(\tau)), x - y(\tau)\rangle \dot{\alpha}(\tau) d\tau + \phi_0(x)\Big)\Big|_{z(t)} &= 0 \\ \Rightarrow \int_0^t \nabla f(y(\tau))\dot{\alpha}(\tau) + \phi_0'(z(t)) &= 0 \\ \Rightarrow \nabla f(y(t))\dot{\alpha}(t) + \phi_0''(z(t))\dot{z}(t) &= 0 \end{aligned}$$

By assuming $\phi_0(x)$ to be quadratic of the form $\phi_0(x) = \frac{1}{2}(x - z(0))^2 + c$, where c is a constant:

$$\dot{z}(t) = -\dot{\alpha}(t)\nabla f(y(t)) \tag{8.3}$$

Let's return to the original condition:

$$\begin{aligned} & f(x(t)) - \phi_t(z(t)) \leq 0 \\ \Leftrightarrow \alpha(t)f(x(t)) - \int_0^t f(y(\tau))\alpha(\tau))d\tau - \int_0^t \nabla f(y(\tau)), z(t) - y(\tau)\rangle \dot{\alpha}(\tau) d\tau - \phi_0(z(t)) \leq 0 \end{aligned}$$

Since this inequality is trivially true for $t = 0$, we will use our freedom to choose $y(t)$ to force the derivative of the left hand side to be negative for all t.

Taking the derivative of the left hand side and taking $y(t) = x(t)$ we find the following condition.

$$\langle \alpha\dot{x} - (z - x)\dot{\alpha}, \nabla f(x)\rangle \leq 0$$

The equality case can be enforced by following the dynamics:

$$\dot{x}(t) = (z(t) - x(t))\frac{\dot{\alpha}(t)}{\alpha(t)}$$

Altogether, we have found a system of differential equations whose solution satisfies $f(x(t)) - f(x_*) = \mathcal{O}(\frac{1}{\alpha(t)})$:

$$\begin{bmatrix}\dot{z}\\ \dot{x}\end{bmatrix} = \begin{bmatrix}-\nabla f(x)\dot{\alpha}\\ (z - x)\frac{\dot{\alpha}}{\alpha}\end{bmatrix}, \tag{8.4}$$

At this point we will make a couple of remarks:

Remark 9. *We believe it is possible to relate the accelerated universal methods, presented in [2], of which Nesterov's Accelerated Gradient is a particular case, to discretizations of 8.4.*

Remark 10. *We can derive a Lyapunov function for the system 8.4:*

$$\begin{aligned}\frac{d}{dt}(\alpha(t)(f(x(t)) - f(x_*)) &= \dot{\alpha}f(x(t)) + \alpha\langle\nabla f(x), \dot{x}\rangle - \dot{\alpha}f(x_*)\\ &= \dot{\alpha}\Big(f(x(t)) - f(x_*) + \langle\nabla f(x), z - x_*\rangle + \langle\nabla f(x), x_* - x\rangle\Big)\\ &\leq \dot{\alpha}\langle\nabla f(x), z - x_*\rangle\end{aligned}$$

where in the last inequality we used the convexity of f.
Now note that the right hand side of the inequality is $-\frac{d}{dt}\frac{1}{2}(z - x_*)^2$, so:

$$\frac{d}{dt}\Big(\alpha(t)(f(x(t)) - f(x_*)) + \frac{1}{2}(z - x_*)^2\Big) \leq 0$$

And so $\alpha(t)(f(x(t)) - f(x_*)) + \frac{1}{2}(z - x_*)^2$ is a Lyapunov function for 8.4. This Lyapunov function is suspiciously similar to the one we have used for the high-resolution heavy-ball dynamics, so with an appropiate change of variables in z the two can probably be related.

Chapter 9

Conclusions and Future Work

The main purpose of this thesis has been to introduce a new technique to discretize dynamical systems that arise as continuous-time limits of discrete-time optimization algorithms. This technique is based on ideas borrowed from event-triggered control. We have suggested some improvements on this technique that can yield an improved convergence rate.

Even though all the algorithms we have introduced have been shown to decay the Lyapunov function at the desired rate at each iteration and don't exhibit Zeno behavior, we haven't been able to give any tight theoretical guarantees on their convergence rate, so the question of whether they achieve acceleration still remains a challenge.

Most of the simulations we have run show that some of the most sophisticated algorithms we have introduced, like the *Adaptive-a* Algorithm or the *Performance-Based Sampled Integrator* are very close to the performance of the standard Polyak's *heavy-ball* for quadratic objectives.

An interesting line of research in that sense is that of combining the algorithms introduced in chapter 7. Hypothetically, the combination of all the improvement directions in a single algorithm might lead to an optimal convergence rate. Once again, proving this is might be hard and further research should be conducted.

As has been noted elsewhere, all the event-triggered algorithms implemented throughout the thesis are *greedy*, in the sense that at each iteration they try to find the maximal stepsize for which the *triggering* condition remains true. This is not necessarily an optimal heuristic, since a more cautious stepsize might lead to improved performance later on. Developing a framework to decide when a certain algorithm should be *greedy* can probably lead to much

better convergence rates.
On the other hand, by reviewing the literature on *high-resolution* differential equations we have seen how the continuous-time perspective can shed some light into the phenomenon of acceleration, and why Nesterov's Accelerated Gradient is superior to Polyak's Heavy Ball for generic convex functions. The success of this approach suggests that the ideas introduced in chapter 8, by considering the continuous-time analogue of Nesterov's *estimating sequences*, might also be a promising line of research to understand the fundamental properties of acceleration and the difference between optimization algorithms.

Bibliography

[1] A. C. Wilson A. Wibisono and M. I. Jordan. "A variational perspective on accelerated methods in optimization". In: *Proceedings of the National Academy of Sciences* ().

[2] B. Recht A. Wilson and M. Jordan. "A Lyapunov Analysis of Momentum Methods in Optimization". In: *arXiv preprint* (2016).

[3] Z. Allen-Zhu and L. Orecchia. "Linear Coupling: An Ultimate Unification of Gradient and Mirror Descent". In: *arXiv preprint* (2016).

[4] M. Jordan B. Shi S. Du and W. Su. "Acceleration via symplectic discretization of high-resolution differential equations". In: *33rd Conference on Neural Information Processing Systems (NeurIPS 2019)* (2019).

[5] M. Jordan B. Shi S. Du and W. Su. "Understanding the Acceleration Phenomenon via High-Resolution Differential Equations". In: *arXiv preprint* ().

[6] E. Garcia C. Nowzari and J.Cortés. "Event-Triggered Communication and Control of Networked Systems for Multi-Agent Consensus". In: *Automatica* (2019).

[7] U. Helmke and J. Moore. *Optimization and Dynamical Systems*. 1996.

[8] S. Sra J. Zhang A. Mokhtari and A. Jadbabaie. "Direct Runge-Kutta discretization achieves acceleration". In: *32nd Conference on Neural Information Processing Systems (NeurIPS 2018)* (2018).

[9] B.Recht L. Lessard and A. Packard. "Analysis and Design of Optimization Algorithms via Integral Quadratic Contraints". In: *arXiv preprints* (2014).

[10] M. Muehlebach and M. I. Jordan. "A Dynamical Systems Perspective on Nesterov Acceleration". In: *arXiv preprint* (2019).

[11] Yurii Nesterov. "A method of solving a convex programming problem with convergence rate $\mathcal{O}(1/k^2)$". In: *Soviet Mathematics Doklady* (1983).

[12] Yurii Nesterov. *Lectures on Convex Optimization*. 2018.

[13] Boris T Polyak. "Some methods of speeding up the convergence of iteration methods". In: *Computational Mathematics and Mathematical Physics* (1964).

[14] Benjamin Recht. "Lyapunov analysis and the Heavy Ball Method". In: (2012). URL: `http://pages.cs.wisc.edu/~brecht/cs726docs/HeavyBallLinear.pdf`.

[15] Y. T. Lee S. Bubeck and M. Singh. "A geometric alternative to Nesterov's accelerated gradient descent". In: *arXiv preprint* (2015).

[16] M. Vaquero and J. Cortés. "Convergence-Rate-Matching Discretization of Accelerated Optimization Flows Through Opportunstic State-Triggered Control". In: *33rd Conference on Neural Information Processing Systems (NeurIPS 2019)* (2019).

[17] Boyd W. Su S and E. Candès. "A Differential Equation for Modeling Nesterov's Accelerated Gradient Method: Theory and Insights". In: *Journal of Machine Learning Research 17 (2016) 1-43* (2015).

[18] Xingyu Zhou. "On the Fenchel Duality between Strong Convexity and Lipschitz Continuous Gradient". In: *arXiv preprints* ().

Appendices

Appendix A

Smooth and Strongly Convex Functions

In this appendix we present some properties of smooth and strongly-convex functions, which are the type of functions we use in most of the thesis. This properties are used in some of the proofs throughout the thesis.

Definition 37. *A differentiable function f is μ-strongly-convex if:*

$$f(y) \geq f(x) + \nabla f(x)^T (y - x) + \frac{\mu}{2} \|y - x\|^2 \tag{A.1}$$

for some $\mu > 0$ and all x, y

Proposition 38 ([18])**.** *The following conditions are all equivalent to a differentiable function f being μ-strongly-convex:*

1. $f(y) \geq f(x) + \nabla f(x)^T (y - x) + \frac{\mu}{2} \|y - x\|^2$;
2. $g(x) = f(x) - \frac{\mu}{2} \|x\|^2$ *is convex;*
3. $(\nabla f(x) - \nabla f(y))^T (x - y) \geq \mu \|x - y\|^2$;
4. $f(\alpha x + (1 - \alpha) y) \leq \alpha f(x) + (1 - \alpha) f(y) - \frac{\alpha(1-\alpha)\mu}{2} \|x - y\|^2 \;\; \alpha \in [0, 1]$;

$\forall\, x, y$

Proof. $1 \equiv 2$: It follows from the first-order condition for convexity of $g(x)$, i.e, $g(x)$ is convex if and only if $g(y) \geq g(x) + \nabla g(x)^T (y - x)\ \forall x, y$

$2 \equiv 3$: It follows from the monotone gradient condition for convexity of $g(x)$, i.e., $g(x)$ is convex if and only if $(\nabla g(x) - \nabla g(y))^T(x-y) \geq 0 \ \forall x, y$

$2 \equiv 4$: It follows from the definition of convexity: $g(x)$ is convex if $g(\alpha x + (1-\alpha)y) \leq \alpha g(x) + (1-\alpha)g(y) \ \forall x, y, \alpha \in [0,1]$

□

Now we will list some conditions that are implied by strong convexity but which are not equivalent to it:

Proposition 39 ([18])**.** *Let f be a continuously differentiable function. The following conditions are all implied by strong convexity:*

1. $\frac{1}{2}\|\nabla f(x)\|^2 \geq \mu(f(x) - f(x_*)) \ \forall x$
2. $\|\nabla f(x) - \nabla f(y)\| \geq \mu \|x - y\| \ \ \forall x, y$
3. $f(y) \leq f(x) + \nabla f(x)^T(y-x) + \frac{1}{2\mu}\|\nabla f(y) - \nabla f(x)\|^2 \ \ \forall x, y$
4. $(\nabla f(x) - \nabla f(y))^T(x-y) \leq \frac{1}{\mu}\|\nabla f(x) - \nabla f(y)\|^2 \ \ \forall x, y$

Proof. 1. This is often called the Polyak-Lojasiewicz (PL) inequality. Follows from minimizing with respect to y in both sides of the inequality (A.1). The left hand side follows from the definition of minimizer. For the right hand side, it's easy to see that the minimum is achieved for $y = x - \frac{1}{\mu}\nabla f(x)$ and then the result follows.

2. Follows from using Cauchy-Schwartz on the characterization 3 (in the last Proposition) of strong-convexity.

3. Define the function $\phi_x(z) = f(z) - \nabla f(x)^T z$. It is easy to see that $\phi_x(z)$ is strongly-convex with the same μ since

$$(\nabla \phi_x(z_1) - \nabla \phi_x(z_2))^T(z_1 - z_2) = (\nabla f(z_1) - \nabla f(z_2))^T(z_1 - z_2) \geq \mu \|z_1 - z_2\|^2$$

where we have again used the characterization 3 from the last Proposition. Now the result follows from applying the Polyak-Lojasiewicz inequality to the strongly-convex function $\phi_x(z)$ (taking into account that $z_* = x$)-

4. Interchanging x and y in the previous condition:

$$f(x) \leq f(y) + \nabla f(y)^T(x-y) + \frac{1}{2\mu}\|\nabla f(x) - \nabla f(y)\|^2$$

We get the result we are looking for by adding up this inequality with the characterization 3 of the previous Proposition.

□

Now let's define L-smooth function:

Definition 40. *A function f is L-smooth, or has L-Lipschitz continuous gradient if:*

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x-y\|$$

Proposition 41. *Let f be a convex function with L-Lipschitz continuous gradient over $\mathbb{R}^n$. Then the following conditions are equivalent:*

1. $\|\nabla f(x) - \nabla f(y)\| \leq L\|x-y\|, \ \forall x, y;$
2. $g(x) = \frac{L}{2}x^Tx - f(x)$ *is convex*
3. $f(y) \leq f(x) + \nabla f(x)^T(y-x) + \frac{L}{2}\|y-x\|^2, \ \forall x, y$
4. $(\nabla f(x) - \nabla f(y))^T(x-y) \leq L\|x-y\|^2, \ \forall x, y$
5. $f(\alpha x + (1-\alpha)y) \geq \alpha f(x) + (1-\alpha)f(y) - \frac{\alpha(1-\alpha)L}{2}\|x-y\|^2, \ \forall x, y, \alpha \in [0,1]$
6. $f(y) \geq f(x) + \nabla f(x)^T(y-x) + \frac{1}{2L}\|\nabla f(y) - \nabla f(x)\|^2, \ \forall x, y$
7. $(\nabla f(x) - \nabla f(y))^T(x-y) \geq \frac{1}{L}\|\nabla f(x) - \nabla f(y)\|$
8. $f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y) - \frac{\alpha(1-\alpha)}{2L}\|\nabla f(x) - \nabla f(y)\|^2, \ \forall x, y, \alpha \in [0,1]$

Proof. $2 \Leftrightarrow 3$: Follows from the first order equivalence of convexity: if $g(x)$ is convex, then $g(y) \geq g(x) + \nabla g(x)^T(y-x)$
$2 \Leftrightarrow 4$: Follows from the fact that if $g(x)$ is convex, then $(\nabla g(x) - \nabla g(y))^T(x-y) \geq 0$

$2 \Leftrightarrow 5$: Follows from the definition of convexity
$6 \Rightarrow 8$: Let $x_\alpha := \alpha x + (1-\alpha)y$, then:

$$f(x) \geq f(x_\alpha) + \nabla f(x_\alpha)^T (x - x_\alpha) + \frac{1}{2L} \|\nabla f(x) - \nabla f(x_\alpha)\|^2$$
$$f(y) \geq f(x_\alpha) + \nabla f(x_\alpha)^T (y - x_\alpha) + \frac{1}{2L} \|\nabla f(y) - \nabla f(x_\alpha)\|^2$$

Multiplying the first inequality by α and the second by $1-\alpha$ and adding them together we get the result after using $\alpha \|x\|^2 + (1-\alpha) \|y\|^2 \geq \alpha(1-\alpha) \|x - y\|^2$
$8 \Rightarrow 6$: We can rewrite the inequality in 8 as:

$$f(y) \geq f(x) + \frac{f(x + \alpha(y-x)) - f(x)}{\alpha} + \frac{1-\alpha}{2L} \|\nabla f(x) - \nabla f(y)\|^2$$

By letting $\alpha \to 0$, we get the inequality in 6.
$6 \Rightarrow 7$: Interchanging x and y we get the following two inequalities:

$$f(y) \geq f(x) + \nabla f(x)^T (y - x) + \frac{1}{2L} \|\nabla f(y) - \nabla f(x)\|^2$$
$$f(x) \geq f(y) + \nabla f(y)^T (x - y) + \frac{1}{2L} \|\nabla f(x) - \nabla f(y)\|^2$$

Now by adding together the two inequalities we get the result.
$7 \Rightarrow 1$: Follows from the Cauchy-Schwartz inequality.
$1 \Rightarrow 4$: Also follows directly from the Cauchy-Schwartz inequality.
$4 \Rightarrow 6$: For this one we need f to be convex. Define $\phi_x(z) = f(z) - \nabla f(x)^T z$. Since f is convex, $\phi_x(z)$ reaches its minimum at $z_* = x$. 4 implies that,

$$(\nabla \phi_x(z_1) - \nabla \phi_x(z_2))^T (z_1 - z_2) \leq L \|z_1 - z_2\|^2$$

which implies that

$$\phi_x(z) \leq \phi_x(y) + \nabla \phi_x(y)^T (z - y) + \frac{L}{2} \|z - x\|^2$$

By minimizing over z, we get 6. $\square$

Appendix B

Computations of Propositions 27 and 31

B.1 Computations for 27

• **Term$^{\mathbf{S}}$ I**. By direct computation we have

$$\nabla V(p) = \begin{bmatrix} (1+\sqrt{\mu s})\nabla f(x) + \sqrt{\mu}v + 2\mu(x - x_*) \\ v + \sqrt{\mu}(x - x_*) \end{bmatrix},$$

and

$$\nabla V(p + tX^a_{\text{hb}}(p)) = \begin{bmatrix} (1+\sqrt{\mu s})\nabla f(x+tv) + \sqrt{\mu}v - t\sqrt{\mu}(1+\sqrt{\mu s})\nabla f(x+av) + 2\mu(x - x_*) \\ v - t\sqrt{\mu}v - t(1+\sqrt{\mu s})\nabla f(x+av) + \sqrt{\mu}(x - x_*) \end{bmatrix}.$$

Therefore,

$$\langle \nabla V(p + tX_{\text{hb}}(p)) - \nabla V(p), X^a_{\text{hb}}(p)\rangle = (1+\sqrt{\mu s})\langle \nabla f(x+tv) - \nabla f(x), v\rangle$$
$$+t2\sqrt{\mu}(1+\sqrt{\mu s})\langle \nabla f(x+av), v\rangle + t2\mu \left\|v\right\|^2 + t(1+\sqrt{\mu s})^2 \left\|\nabla f(x+av)\right\|^2.$$

The RHS of the last expression is precisely $A^S_{ET}(p,t)$. We can use the Lipschitz continuity of the gradient to get $A^S_{ST}(p)$, i.e, the property:

$$\langle \nabla f(x+tv) - \nabla f(x), v\rangle = Lt \left\|v\right\|^2$$

- **Term$^{\mathbf{S}}$ II** We have

$$
\begin{aligned}
&V(p + tX^a_{\mathrm{hb}}(p)) - V(p) = (1 + \sqrt{\mu s})(f(x + tv) - f(x_*)) \\
&+\tfrac{1}{4} \left\| v - t2\sqrt{\mu}v - t(1 + \sqrt{\mu s})\nabla f(x + av) \right\|^2 \\
&+\tfrac{1}{4} \left\| v - t2\sqrt{\mu}v - t(1 + \sqrt{\mu s})\nabla f(x + av) + 2\sqrt{\mu}(x + tv - x_*) \right\|^2 \\
&-(1 + \sqrt{\mu s})(f(x) - f(x_*)) - \tfrac{1}{4} \left\| v \right\|^2 - \tfrac{1}{4} \left\| v + 2\sqrt{\mu}(x - x_*) \right\|^2 .
\end{aligned}
$$

Using $\|a + b\|^2 = \|a\|^2 + 2\langle a, b\rangle + \|b\|^2$ on the second and third addends, the last expression becomes

$$
\begin{aligned}
&(1 + \sqrt{\mu s})(f(x + tv) - f(x)) \underbrace{+\frac{1}{4} \left\| v \right\|^2}_{1} + \tfrac{1}{4} \left\| -t2\sqrt{\mu}v - t(1 + \sqrt{\mu s})\nabla f(x + av) \right\|^2 \\
&+\tfrac{2}{4}\langle v, -t2\sqrt{\mu}v - t(1 + \sqrt{\mu s})\nabla f(x + av)\rangle \underbrace{+\frac{1}{4} \left\| v + 2\sqrt{\mu}(x - x_*) \right\|^2}_{2} \\
&+\tfrac{1}{4} \left\| -t(1 + \sqrt{\mu s})\nabla f(x + av) \right\|^2 + \tfrac{2}{4}\langle v + 2\sqrt{\mu}(x - x_*), -t(1 + \sqrt{\mu s})\nabla f(x + av)\rangle \\
&\underbrace{-\frac{1}{4} \left\| v \right\|^2}_{1} \underbrace{-\frac{1}{4} \left\| v + 2\sqrt{\mu}(x - x_*) \right\|^2}_{2} .
\end{aligned}
$$

Canceling out the corresponding terms (1 and 2) we get

$$
\begin{aligned}
&(1 + \sqrt{\mu s})(f(x + tv) - f(x)) + \tfrac{1}{4} \left\| -t2\sqrt{\mu}v - t(1 + \sqrt{\mu s})\nabla f(x + av) \right\|^2 \\
&+\tfrac{2}{4}\langle v, -t2\sqrt{\mu}v - t(1 + \sqrt{\mu s})\nabla f(x + av)\rangle + \tfrac{1}{4} \left\| -t(1 + \sqrt{\mu s})\nabla f(x + av) \right\|^2 \\
&+\tfrac{2}{4}\langle v + 2\sqrt{\mu}(x - x_*), -t(1 + \sqrt{\mu s})\nabla f(x + av)\rangle.
\end{aligned}
$$

Finally, using

$$
\begin{aligned}
f(x + tv) - f(x) &\leq \langle \nabla f(x), tv\rangle + \frac{L}{2} \left\| tv \right\|^2 , \\
\langle x_* - x, \nabla f(x + av)\rangle &= \langle x_* - x - av + av, \nabla f(x + av)\rangle \\
&= \langle x_* - x - av, \nabla f(x + av)\rangle + \langle av, \nabla f(x + av)\rangle \\
&\leq -\frac{\left\| \nabla f(x + av) \right\|^2}{L} + \langle av, \nabla f(x + av)\rangle,
\end{aligned}
$$

we can uncover the expression of BC_{ET}, B_{ST} and C_{ST}.

• **Term$^{\mathbf{S}}$ III**. We have already computed the expression of D^S_{ET} in Theorem 26. We present another bound which, while more difficult to use to our theoretical results, performs better in simulations. We have

$$\begin{aligned}&\langle \nabla V(p), X^a_{\text{hb}}(p)\rangle + \alpha V(p) = (1+\sqrt{\mu s})\langle \nabla f(x), v\rangle + \sqrt{\mu}\left\|v\right\|^2 - 2\sqrt{\mu}\left\|v\right\|^2\\ &-(1+\sqrt{\mu s})\langle \nabla f(x+av), v\rangle - \sqrt{\mu}(1+\sqrt{\mu s})\langle \nabla f(x+av), x - x_*\rangle + \alpha V(x,v)\end{aligned}$$

and using

$$\alpha V(x,v) \leq \alpha(1+\sqrt{\mu s})(f(x) - f(x_*)) + 3/4\alpha\left\|v\right\|^2 + 2\mu\alpha\left\|x - x_*\right\|^2$$

the last expression can be written as

$$\begin{aligned}&-\sqrt{\mu}\left\|v\right\|^2 + \alpha(1+\sqrt{\mu s})(f(x) - f(x_*)) + 3/4\alpha\left\|v\right\|^2 + 2\mu\alpha\left\|x - x_*\right\|^2\\ &+(1+\sqrt{\mu s})\langle \nabla f(x) - \nabla f(x+av), v\rangle - \sqrt{\mu}(1+\sqrt{\mu s})\langle \nabla f(x+av), x - \hat{x}\rangle\end{aligned} \tag{B.1}$$

We use,

$$\sqrt{\mu}(1+\sqrt{\mu s})\langle \nabla f(x+av), x_* - x\rangle = \sqrt{\mu}(1+\sqrt{\mu s})\langle \nabla f(x+av), x_* - x - av + av\rangle.$$

Using strong-convexity the last expression is upper bounded by

$$\begin{aligned}&\sqrt{\mu}(1+\sqrt{\mu s})(f(x_*) - f(x+av))\\ &+\sqrt{\mu}(1+\sqrt{\mu s})(-\mu/2\left\|x + av - x_*\right\|^2)\\ &+\sqrt{\mu}(1+\sqrt{\mu s})\langle \nabla f(x+av), av\rangle.\end{aligned}$$

So equation B.1 reads

$$\begin{aligned}&-\sqrt{\mu}\left\|v\right\|^2 + \alpha(1+\sqrt{\mu s})(f(x) - f(x_*)) + 3/4\alpha\left\|v\right\|^2 + 2\mu\alpha\left\|x - x_*\right\|^2\\ &+(1+\sqrt{\mu s})\langle \nabla f(x) - \nabla f(x+av), v\rangle + \sqrt{\mu}(1+\sqrt{\mu s})(f(x_*) - f(x+av))\\ &+\sqrt{\mu}(1+\sqrt{\mu s})(-\mu/2\left\|x + av - x_*\right\|^2) + \sqrt{\mu}(1+\sqrt{\mu s})\langle \nabla f(x+av), av\rangle,\end{aligned}$$

which re-grouping terms becomes

$$\begin{aligned}&\sqrt{\mu}\left\|v\right\|^2 + \underbrace{\alpha(1+\sqrt{\mu s})(f(x) - f(x_*)) + \sqrt{\mu}(1+\sqrt{\mu s})(f(x_*) - f(x+av))}_{\text{TermI}}\\ &+3/4\alpha\left\|v\right\|^2 + (1+\sqrt{\mu s})\langle \nabla f(x) - \nabla f(x+av), v\rangle\\ &\underbrace{+2\mu\alpha\left\|x - x_*\right\|^2 + \sqrt{\mu}(1+\sqrt{\mu s})(-\mu/2\left\|x + av - x_*\right\|^2)}_{\text{Term II}}\\ &+\sqrt{\mu}(1+\sqrt{\mu s})\langle \nabla f(x+av), av\rangle.\end{aligned} \tag{B.2}$$

We observe that

$$\text{Term I} \leq (1+\sqrt{\mu s})(\alpha(f(x)-f(x_*))+\sqrt{\mu}(f(x_*)-f(x+av)+f(x)-f(x)))$$
$$=(1+\sqrt{\mu s})\big((\alpha-\sqrt{\mu})(f(x)-f(x_*)+\sqrt{\mu}(f(x)-f(x+av)))\big)$$

$$\text{Term II} \leq 2\mu\alpha\left\|x-x_*\right\|^2-\sqrt{\mu}\tfrac{\mu}{2}(1+\sqrt{\mu s})(2\mu\alpha-\sqrt{\mu}(1+\sqrt{\mu s})\tfrac{\mu}{2})\left\|x-x_*\right\|^2$$
$$+2\sqrt{\mu}(1+\sqrt{\mu s})\tfrac{\mu}{2}\left\|x-x_*\right\|\left\|av\right\|-\sqrt{\mu}(1+\sqrt{\mu s})\mu/2\left\|av\right\|$$

and finally resorting to the inequalities

$$f(x_*)-f(x)\leq\frac{-\left\|\nabla f(x)\right\|^2}{2L}$$
$$-\left\|x-x_*\right\|^2\leq-\frac{-\left\|\nabla f(x)\right\|}{L^2}$$
$$\left\|x-x_*\right\|\leq\frac{\left\|\nabla f(x)\right\|}{\mu}$$

equation B.2 becomes

$$(3/4\alpha-\sqrt{\mu})\left\|v\right\|^2+(1+\sqrt{\mu s})(\alpha-\sqrt{\mu})\frac{\left\|\nabla f(x)\right\|^2}{2L}$$
$$+(1+\sqrt{\mu s})\sqrt{\mu}(f(x)+f(x+av))+(2\mu\alpha-\sqrt{\mu}\mu/2(1+\sqrt{\mu s}))\frac{\left\|\nabla f(x)\right\|^2}{L^2}$$
$$+2\sqrt{\mu}\tfrac{\mu}{2}(1+\sqrt{\mu s})\frac{\left\|\nabla f(x)\right\|}{\mu}\left\|av\right\|$$
$$-\sqrt{\mu}(1+\sqrt{\mu s})\mu/2\left\|av\right\|^2-(1+\sqrt{\mu s})\langle\nabla f(x+av)-\nabla f(x),v\rangle+\sqrt{\mu}(1+\sqrt{\mu s})\langle\nabla f(x+av),av\rangle$$

which is the bound for Term^S III given in 27

B.2 Computations of proposition 31

Equalities for *Term I* and II follow from direct computation. Inequality for *Term IV* is derived in [16]. Here we provide a derivation of the bound for *Term III*. By using Holder's inequality appropriately we get:

$$V(p(t))-V(p(0))=(1+\sqrt{\mu s})(f(x(t))-f(\hat{x}))+\frac{1}{2}(\left\|v(t)\right\|^2-\left\|\hat{v}\right\|^2)+$$
$$\frac{1}{2}\left\|a_1(t)+2\sqrt{\mu}a_2(t)\right\|^2+2\langle a_1(t)+2\sqrt{\mu}a_2(t),v+2\sqrt{\mu}(x-x_*)\rangle$$

where $a_1(t) = v(t) - \hat{v}$ and $a_2(t) = x(t) - \hat{x}$

Let's bound now the only term that depends on the minimizer and therefore we don't know how to compute:

$$\begin{aligned}
&\langle a_1(t) + 2\sqrt{\mu}a_2(t), 2\sqrt{\mu}(\hat{x} - x_*)\rangle \\
&= -2\sqrt{\mu}(1 + \sqrt{\mu s})t\langle \nabla f(\hat{x}), \hat{x} - x_*\rangle \\
&\leq 2\sqrt{\mu}t(1 + \sqrt{\mu s})\frac{\|\nabla f(\hat{x})\|^2}{\mu}
\end{aligned}$$

Appendix C

Codes

In this appendix we include some of the Matlab codes written to test some of the algorithms presented during the thesis. The concrete bounds of the triggering conditions used in the codes might differ slightly to the ones presented throughout the thesis but in any case the algorithms are provably convergent and don't suffer from Zeno phenomena.

The following is an implementation of algorithm 4 in the self-triggered case.

```
%% Performance-Based Algorithm
%% Self-Triggered

function [x1valuesst,x2valuesst,normxst,lyapvaluesst
   ,fvaluesst] = performance(f,gradf,optim,mu,L,a,
   lyapunov,Xhba,s,alfa,niter,tol,x0,v0)
    disp('performance')
    xhat = x0;
    vhat = v0;
    phat = [xhat,vhat];
    i = 1;
    while ((norm(gradf(xhat)) > tol) && (i <= niter)
       )
        x1valuesst(i) = xhat(1);
        x2valuesst(i) = xhat(2);
        normxst(i) = norm(xhat(1),xhat(2));
        lyapvaluesst(i) = lyapunov(xhat,vhat);
```

```
15          fvaluesst(i) = f(xhat);
16
17          t1 = (1+sqrt(mu*s))*L*norm(vhat)^2;
18          t2 = 2*sqrt(mu)*(1+sqrt(mu*s))*dot(gradf(
               xhat+a*vhat),vhat);
19          t3 = 2*mu*norm(vhat)^2;
20          t4 = (1+sqrt(mu*s))^2 * norm(gradf(xhat+a*
               vhat))^2;
21          Ast = t1+t2+t3+t4;
22
23          u1 = -alfa*sqrt(mu)*norm(vhat)^2;
24          u2 = -alfa*sqrt(mu)*(1+sqrt(mu*s))/L * norm(
               gradf(xhat+a*vhat))^2;
25          u3 = alfa*sqrt(mu)*(1+sqrt(mu*s))*a*dot(vhat
               ,gradf(xhat+a*vhat));
26          u4 = alfa*(1+sqrt(mu*s))*dot(vhat,gradf(xhat
               )-gradf(xhat+a*vhat));
27          Bst = u1+u2+u3+u4;
28
29          v1 = alfa*(1+sqrt(mu*s))*L/2 *norm(vhat)^2;
30          v2 = alfa/4 *norm(-2*sqrt(mu)*vhat-(1+sqrt(
               mu*s))*gradf(xhat+a*vhat))^2;
31          v3 = alfa/4 * (1+sqrt(mu*s))^2 * norm(gradf(
               xhat+a*vhat))^2;
32          Cst = v1+v2+v3;
33
34          w1 = (1+sqrt(mu*s))*dot(gradf(xhat)-gradf(
               xhat+a*vhat),vhat);
35          w2 = (3*alfa/4 - sqrt(mu)-mu*sqrt(mu)*(1+
               sqrt(mu*s))*a^2 /2)*norm(vhat)^2;
36          w3 = -(mu/2 * sqrt(mu)*(1+sqrt(mu*s))-2*alfa
               *mu)*(1/L^2)*norm(gradf(xhat))^2;
37          w4 = sqrt(mu)*(1+sqrt(mu*s))*a*norm(gradf(
               xhat))*norm(vhat);
38          w5 = a*sqrt(mu)*(1+sqrt(mu*s))*dot(vhat,
               gradf(xhat+a*vhat));
39          w6 = -(1+sqrt(mu*s))*(sqrt(mu)-alfa)/(2*L) *
                norm(gradf(xhat))^2;
```

```
40          w7 = sqrt(mu)*(1+sqrt(mu*s))*(f(xhat)-f(xhat
               +a*vhat));
41          Dst = w1+w2+w3+w4+w5+w6+w7;
42
43          k1 = Cst;
44          k2 = Ast+Bst;
45          k3 = Dst;
46
47          %perfst = @(t)(exp(alfa*t)*(k1/alfa^2 *t^2 +
                (k2/alfa - 2*k1/alfa^2)*t + k3/alfa - k2
               /alfa^2 + 2*k1/alfa^3)-(k3/alfa - k2/alfa
               ^2 + 2*k1/alfa^3));
48          perfst = @(t)integral(@(tau)(exp(alfa*tau)
               .*(k1*tau.^2 + k2*tau + k3)),0,t);
49
50          %to find the interval where the solution is
               we use the fact that
51          %stepst(p) is less than stepstp(p)
52
53          cnt = 2;
54          stepst = (-(Ast+Bst)+sqrt((Ast+Bst)^2 - 4*
               Cst*Dst))/(2*Cst);
55
56          while(perfst(cnt*stepst) < 0)
57              cnt = cnt + 1;
58          end
59
60          tk = fzero(perfst,[stepst,cnt*stepst]);
61
62          phat = phat + tk*Xhba(xhat,vhat,a);
63          xhat = [phat(1),phat(2)];
64          vhat = [phat(3),phat(4)];
65          i = i+1;
66      end
67  end
```

The following is an implementation of algorithm 5 in the self-triggered case.

```
%% Making Use of Sampling Information
%% Self-Triggered Algorithm

function [x1valuesst,x2valuesst,normxst,lyapvaluesst
   ,fvaluesst] = predictedzoh(f,gradf,optim,mu,L,a,
   lyapunov,Xhba,s,alfa,niter,tol,x0,v0)
    disp('predictedzoh')
    xhat = x0;
    vhat = v0;
    phat = [xhat,vhat];
    i = 1;
    while ((norm(gradf(xhat)) > tol) && (i <= niter)
       )
        x1valuesst(i) = xhat(1);
        x2valuesst(i) = xhat(2);
        normxst(i) = norm(xhat(1),xhat(2));
        lyapvaluesst(i) = lyapunov(xhat,vhat);
        fvaluesst(i) = f(xhat);

        t1 = (1+sqrt(mu*s))*L*norm(vhat)^2;
        t2 = 2*sqrt(mu)*(1+sqrt(mu*s))*dot(gradf(
           xhat+a*vhat),vhat);
        t3 = 2*mu*norm(vhat)^2;
        t4 = (1+sqrt(mu*s))^2 * norm(gradf(xhat+a*
           vhat))^2;
        Ast = t1+t2+t3+t4;

        u1 = -alfa*sqrt(mu)*norm(vhat)^2;
        u2 = -alfa*sqrt(mu)*(1+sqrt(mu*s))/L * norm(
           gradf(xhat+a*vhat))^2;
        u3 = alfa*sqrt(mu)*(1+sqrt(mu*s))*a*dot(vhat
           ,gradf(xhat+a*vhat));
        u4 = alfa*(1+sqrt(mu*s))*dot(vhat,gradf(xhat
           )-gradf(xhat+a*vhat));
        Bst = u1+u2+u3+u4;

        v1 = alfa*(1+sqrt(mu*s))*L/2 *norm(vhat)^2;
```

```
30          v2 = alfa/4 *norm(-2*sqrt(mu)*vhat-(1+sqrt(
               mu*s))*gradf(xhat+a*vhat))^2;
31          v3 = alfa/4 * (1+sqrt(mu*s))^2 * norm(gradf(
               xhat+a*vhat))^2;
32          Cst = v1+v2+v3;
33
34          w1 = (1+sqrt(mu*s))*dot(gradf(xhat)-gradf(
               xhat+a*vhat),vhat);
35          w2 = (3*alfa/4 - sqrt(mu)-mu*sqrt(mu)*(1+
               sqrt(mu*s))*a^2 /2)*norm(vhat)^2;
36          w3 = -(mu/2 * sqrt(mu)*(1+sqrt(mu*s))-2*alfa
               *mu)*(1/L^2)*norm(gradf(xhat))^2;
37          w4 = sqrt(mu)*(1+sqrt(mu*s))*a*norm(gradf(
               xhat))*norm(vhat);
38          w5 = a*sqrt(mu)*(1+sqrt(mu*s))*dot(vhat,
               gradf(xhat+a*vhat));
39          w6 = -(1+sqrt(mu*s))*(sqrt(mu)-alfa)/(2*L) *
                norm(gradf(xhat))^2;
40          w7 = sqrt(mu)*(1+sqrt(mu*s))*(f(xhat)-f(xhat
               +a*vhat));
41          Dst = w1+w2+w3+w4+w5+w6+w7;
42
43          tk = (-(Ast+Bst)+sqrt((Ast+Bst)^2 - 4*Cst*
               Dst))/(2*Cst);
44
45          phat = phat + tk*Xhba(xhat,vhat,a);
46          xhat = [phat(1),phat(2)];
47          vhat = [phat(3),phat(4)];
48          i = i+1;
49      end
50  end
```

The following is an implementation of algorithm 6 where at each iteration, we start with a^* and then adaptively find an a suitable for that particular iteration.

```
1  % Adaptive-a algorithm
2  % We start each iteration with a certain $a$
3  %Then we tune this a with an increasing rate and a
```

```
   decreasing rate
% Based on the heuristic "the higher the a, the
   longer the stepsize"
function [x1values, x2values, normx, lyapvalues,
   fvalues] = adaptiveadef(f, gradf, optim, mu, L,
   a0, s, alfa, niter, tol, x0, v0, lyapunov, Xhba,
   epsilon1, epsilon2, increate, decrate)
    disp('adaptiveadef')
    xhat = x0;
    vhat = v0;
    phat = [xhat,vhat];
    i = 1;
    while((norm(gradf(xhat)) > tol) && (i <= niter))
        i
        x1values(i) = xhat(1);
        x2values(i) = xhat(2);
        normx(i) = norm(xhat(1),xhat(2));
        lyapvalues(i) = lyapunov(xhat,vhat);
        fvalues(i) = f(xhat);

        aa = a0;

        t1 = @(a)((1+sqrt(mu*s))*L*norm(vhat)^2);
        t2 = @(a)(2*sqrt(mu)*(1+sqrt(mu*s))*dot(
           gradf(xhat+a*vhat),vhat));
        t3 = @(a)(2*mu*norm(vhat)^2);
        t4 = @(a)((1+sqrt(mu*s))^2 * norm(gradf(xhat
           +a*vhat))^2);
        Ast = @(a)(t1(a)+t2(a)+t3(a)+t4(a));

        u1 = @(a)(-alfa*sqrt(mu)*norm(vhat)^2);
        u2 = @(a)(-alfa*sqrt(mu)*(1+sqrt(mu*s))/L *
           norm(gradf(xhat+a*vhat))^2);
        Bst = @(a)(u1(a)+u2(a));

        v1 = @(a)(alfa*(1+sqrt(mu*s))*L/2 *norm(vhat
           )^2);
        v2 = @(a)(alfa/4 *norm(-2*sqrt(mu)*vhat-(1+
```

```
            sqrt(mu*s))*gradf(xhat+a*vhat))^2);
        v3 = @(a)(alfa/4 * (1+sqrt(mu*s))^2 * norm(
            gradf(xhat+a*vhat))^2);
        Cst = @(a)(v1(a)+v2(a)+v3(a));

        w1 = @(a)((alfa*2*mu-sqrt(mu)*(1+sqrt(mu*s))
            *mu/2)*1/L^2)*norm(gradf(xhat))^2;
        w2 = @(a)((alfa*0.75-sqrt(mu))*norm(vhat)^2)
            ;
        w3 = @(a)((1+sqrt(mu*s))*(alfa-sqrt(mu))/(2*
            L) *norm(gradf(xhat))^2);
        w4 = @(a)(-a*(1+sqrt(mu*s))*mu*norm(vhat)^2)
            ;
        w5 = @(a)(a/sqrt(mu) *(1+sqrt(mu*s))*L*norm(
            vhat)*norm(gradf(xhat)));
        Dst = @(a)(w1(a)+w2(a)+w3(a)+w4(a)+w5(a));

        tk = @(a)((-(Ast(a)+Bst(a))+sqrt((Ast(a)+Bst
            (a))^2 - 4*Cst(a)*Dst(a)))/(2*Cst(a)));
        that = tk(0);

        if((Dst(aa) < 0) && (that/2 < tk(aa)))
            phat = phat + tk(aa)*Xhba(xhat,vhat,aa);
            aa = aa*incrate;
        else
            while((Dst(aa) >= 0) || (that/2 >= tk(aa
                )))
                aa = aa*decrate;
            end
            phat = phat + tk(aa)*Xhba(xhat,vhat,aa);
            aa = aa*incrate;
        end
        phat = phat + tk(aa)*Xhba(xhat,vhat,aa);
        xhat = [phat(1),phat(2)];
        vhat = [phat(3),phat(4)];
        i = i+1;
    end
end
```

The following is an implementation of algorithm 7

```
function [x1values, x2values, normx, lyapvalues,
   fvalues] = fohdefinitiu2(f, gradf, optim, mu, L,
   s, alfa, niter, tol, x0, v0, lyapunov)
    disp('foh-definitiu')
    xhat = x0;
    vhat = v0;
    i = 1;
    while((norm(gradf(xhat)) > tol) && (i <= niter))
        x1values(i) = xhat(1);
        x2values(i) = xhat(2);
        normx(i) = norm(xhat(1),xhat(2));
        lyapvalues(i) = lyapunov(xhat,vhat);
        fvalues(i) = f(xhat);

        K1 = -vhat/(2*sqrt(mu))-(1+sqrt(mu*s))/(4*mu
           )*gradf(xhat);
        K2 = xhat + vhat/(2*sqrt(mu))+(1+sqrt(mu*s))
           /(4*mu)*gradf(xhat);

        xt = @(t)(K1*exp(-2*sqrt(mu)*t)-(1+sqrt(mu*s
           ))/(2*sqrt(mu))*gradf(xhat)*t + K2);
        vt = @(t)(-2*sqrt(mu)*K1*exp(-2*sqrt(mu)*t)
           -(1+sqrt(mu*s))/(2*sqrt(mu))*gradf(xhat))
           ;
        a1 = @(t)(vt(t)-vhat);
        a2 = @(t)(xt(t)-xhat);

        q1 = @(t)((1+sqrt(mu*s))*dot(gradf(xt(t))-
           gradf(xt(0)),vt(t)));
        q2 = @(t)(-sqrt(mu)*dot(vt(t)-vhat,vt(t)));
        q3 = @(t)(-(1+sqrt(mu*s))*dot(vt(t)-vhat,
           gradf(xhat)));
        q4 = @(t)(-sqrt(mu)*(1+sqrt(mu*s))*dot(xt(t)
           -xhat,gradf(xhat)));

```

```
26          termh1 = @(t)(q1(t)+q2(t)+q3(t)+q4(t));
27          %termh1(0)
28
29
30          w1 = @(t)((1+sqrt(mu*s))*dot(gradf(xhat),vt(
               t)-vhat));
31          w2 = @(t)(-sqrt(mu)*dot(vhat,vt(t)-vhat));
32
33          termh2 = @(t)(w1(t)+w2(t));
34          %termh2(0)
35
36          r1 = @(t)(alfa*(1+sqrt(mu*s))*(f(xt(t))-f(
               xhat)));
37          r2 = @(t)(alfa/4 *(norm(vt(t))^2-norm(vhat)
               ^2));
38          r3 = @(t)(alfa/4 * norm(a1(t)+2*sqrt(mu)*a2(
               t))^2);
39          r4 = @(t)(alfa/2 * dot(a1(t)+2*sqrt(mu)*a2(t
               ),vhat));
40          r5 = @(t)(alfa*(1+sqrt(mu*s))/(2*mu) *t*norm
               (gradf(xhat))^2);
41
42          termh3 = @(t)(r1(t)+r2(t)+r3(t)+r4(t)+r5(t))
               ;
43          %termh3(0)
44
45          termh4 = (alfa*3/4 - sqrt(mu))*norm(vhat)^2
               + ((1+sqrt(mu*s))*(alfa-sqrt(mu))/(2*L) +
                (alfa*2*mu-sqrt(mu)*(1+sqrt(mu*s))*mu/2)
               *1/L^2)*norm(gradf(xhat))^2;
46          %termh4
47
48          gET = @(t)(termh1(t)+termh2(t)+termh3(t)+
               termh4);
49
50          right = 0.5;
51          while(gET(right) < 0)
52              right = right*1.5;
```

```
53          end
54
55          %gET(0)
56          %gET(right)
57
58          tk = fzero(gET,[0,right]);
59
60          xhat = xt(tk);
61          vhat = vt(tk);
62          i = i+1;
63      end
64  end
```

List of Figures