Final report

June 20, 2022

과제학생: 전기정보공학부 2017-10673 김성윤
과제명: 미분방정식(또는 제어이론)을 이용한 최적화 알고리즘 개발
담당교수: 양인순 교수님
학생 서명: 김성윤

.

1 Introduction

In this report I aim to show the motives and results of my graduate project. The report is divided into four sections. In the Backgrounds section I explain the results of [1], [2], [3], which are intimately related to the project. In the continuous case analysis section, I study the general property of the first order system

$$\ddot{X} = -\nabla f(X) \tag{1}$$

$$\dot{X} = 0 \text{ if } \langle \nabla f(X), \dot{X} \rangle < 0 \tag{2}$$

and prove convergence results when f is quadratic or strongly convex. In the discrete case analysis section, I study the convergence of the discrete optimization algorithm motivated by the continuous case differential equation, which is equivalent to using gradient restarting for Nesterov's accelerated gradient descent algorithm with large constant momentum. At last, in the Numerical analysis section I show some numerical proof that using constant large momentum can speed up convergence in multiple cases.

2 Backgrounds

2.1 A continuous counterpart of Nesterov's algorithm

In convex optimization, where the task is minimizing a convex function $f(x)$ on a domain Ω of f, it is proven that the lower bound of the convergence rate of first order optimization algorithms is given as $O(\frac{1}{k^2})$. In other words, for any first order optimization algorithm on a class of convex functions, the general convergence rate holding for all convex f cannot be better than $O(\frac{1}{k^2})$. Vanila gradient descent has the convergence rate of $O(\frac{1}{k})$, so there is a gap between the lower bound and the actual convergence rate. Nesterov filled the gap with his celebrated Nesterov's algorithm, which is proven to have $O(\frac{1}{k^2})$ convergence rate for any convex f. In this sense, Nesterov's algorithm is often

said to have optimal convergence rate for convex optimization. The iteration of Nesterov's algorithm is given as

$$x_{k+1} = y_k - \frac{1}{L}\nabla f(y_k) \tag{3}$$

$$y_{k+1} = x_{k+1} + \theta_k(x_{k+1} - x_k) \tag{4}$$

where θ_k satisfies

$$\theta_k = \frac{\alpha_k(1-\alpha_k)}{{\alpha_k}^2 + \alpha_{k+1}} \tag{5}$$

and α_k is defined by the iteration

$$\alpha_{k+1}^2 = (1-\alpha_{k+1})\alpha_k^2 \tag{6}$$

for arbitrary α_0 between 0 and 1([4]). Here, L is the lipschitz constant of ∇f.

When we calculate θ_k, it converges to $\frac{k-1}{k+2}$ after a few steps: Thus, we can use the simplified iteration as a substitute for Nesterov's algorithm:

$$x_{k+1} = y_k - \alpha\nabla f(y_k) \tag{7}$$

$$y_{k+1} = x_{k+1} + \frac{k-1}{k+2}(x_{k+1} - x_k) \tag{8}$$

Here, α is the learning rate of the optimization algorithm, which is a hyperparameter chosen to be less than $\frac{1}{L}$.

In [1], the authors discuss the continuous time counterpart of this simplified iteration. The continuous time version of the algorithm can be found first by first making the iteration as an expression only consisting of ys, and taking the limit of α to 0. A rigorous proof that the discrete algorithm actually follows the trajectory of the found continuous system is given in [1]. Here, I will only show the intuition behind it: By plugging in $x_{k+1} = y_k - \alpha\nabla f(y_k)$ and $x_k = y_{k-1} - \alpha\nabla f(y_{k-1})$, we get

$$y_{k+1} = y_k - \alpha\nabla f(y_k) + \frac{k-1}{k+2}(y_k - \alpha\nabla f(y_k) - y_{k-1} + \alpha\nabla f(y_{k-1}) \tag{9}$$

After some calculation we get

$$y_{k+1} - 2y_k + y_{k-1} + \frac{3}{k+2}(y_k - y_{k-1}) + \alpha\nabla f(y_k) + \frac{3}{k+2}\alpha(\nabla f(y_k) - f(y_{k-1})) = 0 \tag{10}$$

and dividing the equation to α and taking the limit on both sides yields

$$\ddot{Y} + \frac{3}{t}\dot{Y} + \nabla f(Y) = 0 \tag{11}$$

when α is very small there is no difference between x and y, and we may write

$$\ddot{X} + \frac{3}{t}\dot{X} + \nabla f(X) = 0 \tag{12}$$

To summarize, Nesterov's momentum algorithm follows the trajectory of a second-order system, and the momentum coefficient $\frac{k-1}{k+2}$ is analogous to the damping coefficient $\frac{3}{t}$.

Also, the authors in [1] discuss a more generalized differential equation of the form

$$\ddot{X} + \frac{r}{t}\dot{X} + \nabla f(X) = 0 \tag{13}$$

and its discrete version (as a first order optimization algorithm) given as

$$x_{k+1} = y_k - \alpha \nabla f(y_k) \tag{14}$$

$$y_{k+1} = x_{k+1} + \frac{k-1}{k+r-1}(x_{k+1} - x_k) \tag{15}$$

With different values of r the convergence pattern of each optimization algorithm varies. For r smaller than 3, the continuous dynamics may not converge to a single point. For r larger than 3, when r is small it converges fast at first because of low damping, but converges slower at the latter phase of the algorithm because there is so little damping. In contrast, when r is large it converges slow at first, but converges faster than small r in the end. Thus, selecting appropriate r matters for the algorithm to converge faster. For details, see figure 1.

These experiments give arise to a natural question: for a general second-order system given as

$$\ddot{X} + r(t)\dot{X} + \nabla f(X) = 0 \tag{16}$$

what would be the optimal damping coefficient $r(t)$ that would make the optimization algorithm converge the fastest? Or, how could we manipulate $r(t)$ to make the optimization algorithm converge faster, at least for certain family of functions? These were the questions that I aimed to answer for the graduate project.

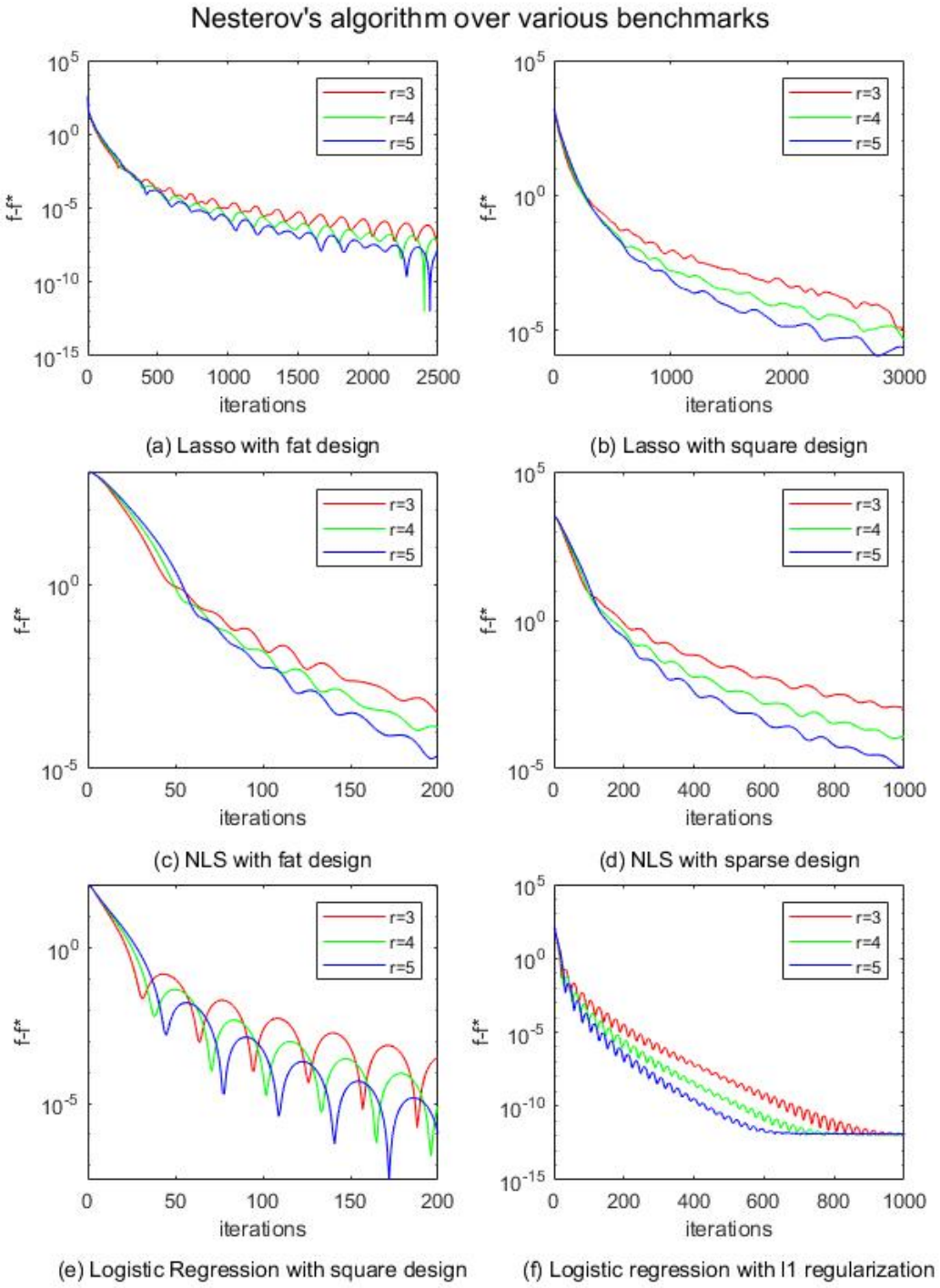


Figure 1: Nesterov's algorithm over various benchmarks, varying r

2.2 Adaptive restarting to speed up convergence of Nesterov's algorithm

In [2], the authors present "adaptive restarting", a variant of Nesterov's algorithm. The basic idea of adaptive restarting is speeding up convergence of the original algorithm by erasing overshoots that happen in Nesterov's algorithm. Nesterov's algorithm is momentum-based, so when the momentum is accumulated, x_k may overshoot as in figure 2. By restarting the algorithm, i.e. by making $x_{k+1} = y_{k+1} = x_k$ when overshooting happens at step k, [2] successfully decreases overshooting in Nesterov's algorithm, leading to a faster convergence.

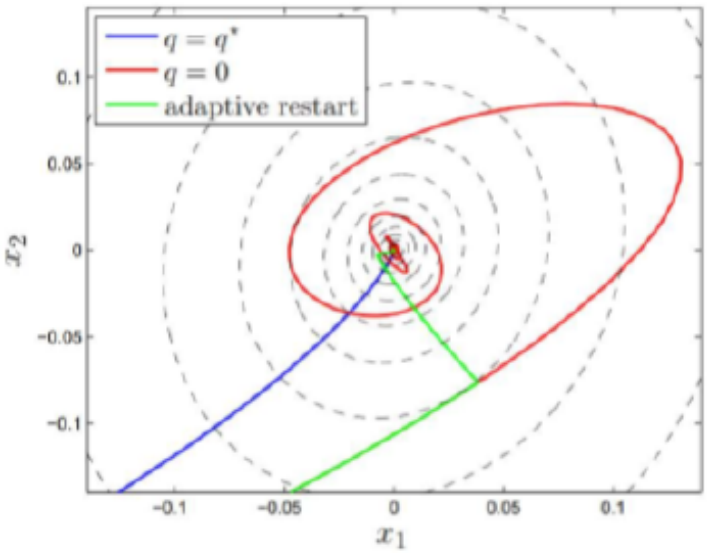


Figure 2: Restarting methods help decrease overshoots([2])

Two specific algorithms are presented in [2], the functional restarting scheme and the gradient restarting scheme. The functional restarting scheme compares consecutive values of the objective function to check if overshooting is happening, and the gradient restarting scheme calculates $\langle \nabla f(y_{k-1}), x_k - x_{k-1} \rangle$ and restarts if the value is larger than 0. Both methods work fairly in most practical problems, though the functional restarting scheme has an additional overhead as it should find actual values of the objective function at each iteration.

These methods can be understood as choosing an appropriate damping coefficient at each step of the optimization algorithm. Specifically, we can think of the algorithm as θ_k being $\frac{k_r - 1}{k_r + 2}$, where k_r denotes the consecutive steps before the latest restart + 1. Thus, they can be understood as examples of choosing a series of damping coefficients that efficiently decrease overshooting of Nesterov's algorithm.

2.3 Convergence rate of restarting based gradient algorithm

In [1] the authors claim that the gradient restarting algorithm is yet proven to have linear convergence. Actually, in [3], linear convergence of a certain type of gradient restarting scheme is proven. Specifically, the author proves linear convergence of the algorithm when f is a strongly convex func-

tion:

Algorithm 1 Generalized Restarted Momentum Method([3])

Require: x_0, $x_{-1} = x_0$
for $i = 0, 1, 2, ...$ **do**
  $z_{k+1} = \beta_k(x_k - x_{k-1}) - \alpha_k \nabla f(x_k)$
  if $\langle f(x_k + z_{k+1}), z_{k+1} \rangle > 0$ **then**
    $x_{k+1} = x_k - \alpha_k \nabla f(x_k)$
  else
    $x_{k+1} = x_k + z_{k+1}$
  end if
end for

This algorithm can be thought of as gradient restarting algorithm applied on Heavy Ball algorithm. After proving linear convergence for algorithm 1, the authors also provide a sketch of proof to extend the original proof in the case of Nesterov iteration. This result is meaningful because it shows that momentum restarting techniques are schemes that can guarantee linear convergence of the algorithm, not knowing the strong convex parameter.

3 Continuous Case Analysis

3.1 The second order ODE

From the background we could understand two things: First, for the dynamics

$$\ddot{X} + r(t)\dot{X} + \nabla f(X) = 0 \tag{17}$$

the choice of $r(t)$ may influence the motion and the corresponding discrete system. Second, gradient restarting methods is useful to prevent overshooting of momentum algorithms.

When it comes to designing an effective $r(t)$ that will speed up convergence, one can easily come up with erasing the damping and making it constantly zero. However, it is clear that a dynamics with no friction may fail in converging: A simple example is the harmonic system $\ddot{x} + x = 0$, where the system oscillates in an infinite manner. Hence, a motivation of "restarting" is natural, for the system to prevent oscillating infinitely and finally converge.

In this sense, we study the convergence of the second-order dynamics

$$\ddot{X} = -\nabla f(X) \tag{18}$$

$$\dot{X} = 0 \text{ if } \langle \nabla f(X), \dot{X} \rangle < 0 \tag{19}$$

3.2 Pontryagin's maximal principle and optimal control for the system

In this section we will see that our choice of $r(t)$ made in section 3.1 is in a family of functions that suffices the necessary condition to be optimal.

Pontryagin's maximal principle provides a necessary condition for optimal control in a continuous dynamic system. It is given as follows:

Lemma 1 (Pontryagin's maximal principle)**.** *Let $x \in \mathbb{R}^d$ be the state of the dynamic system*

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

, and we want to find the control u that minimizes the cost function

$$J[x(t)] = \int_0^T L(x, u, t)\, dt \ + \psi(x(T)) \tag{20}$$

along the trajectory. Let the Hamiltonian of the system be

$$H(t, x, u, \lambda) = \langle \lambda, F(x, u, t) \rangle + L(x, u, t) \tag{21}$$

Then, when we denote the optimal control as u^* and the corresponding trajectory as x^*, there exists a costate function $\lambda^* : [0, T] \to \mathbb{R}^d$ that satisfies the followings:

$$\dot{x^*} = \nabla_\lambda H(t, x^*, u^*, \lambda^*) \tag{22}$$

$$\dot{\lambda^*} = -\nabla_x H(t, x^*, u^*, \lambda^*) \tag{23}$$

$$H(t, x^*, u^*, \lambda^*) = \min_u H(t, x^*, u, \lambda^*) \tag{24}$$

and

$$\lambda(T) = \nabla \psi(x^*(T)) \tag{25}$$

We can write our system $X'' + u(t)X' + \nabla f(X) = 0$ as a linear system as below:

$$\begin{bmatrix} X_1 \\ X_2 \end{bmatrix}' = \begin{bmatrix} O & I \\ O & u(t)I \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} - \begin{bmatrix} O \\ \nabla f(X_1) \end{bmatrix} \tag{26}$$

where X_1, X_2 are vectors in $\mathbb{R}^d$ and $u(t)$ is the momentum coefficient(or the control) of the system. Also, we can think of the cost function of the system as:

$$J[x(t)] = \int_0^T \langle \nabla f(X_1), X_2 \rangle\, dt \ = f(X_1(T)) - f(X_1(0)) \tag{27}$$

which holds as $\dot{X_1} = X_2$. It is obvious that we want to find the control that minimizes the cost function in (27), as the objective of the optimization algorithm(or its continuous counterpart) is to decrease the function value $f(X_1)$.

Now, we can apply Pontryagin's principle to the system to find the necessary conditions for optimal $u(t)$. Suppose $u(t)$ is bounded in $[a, b]$, where a, b are real numbers. The optimal control should be a minimizer to the Hamiltonian

$$H(t, x^*, u, \lambda^*) = \langle \lambda_1^*, X_2 \rangle - u(t) \langle \lambda_2^*, X_2 \rangle - \langle \lambda_2^*, \nabla f(X_1) \rangle + \langle \nabla f(X_1), X_2 \rangle \tag{28}$$

meaning that it should be given as a discontinuous control, where

$$u^*(t) = \begin{cases} a & \text{when} \quad \langle \lambda_2^*, X_2 \rangle > 0 \\ b & \text{when} \quad \langle \lambda_2^*, X_2 \rangle < 0 \end{cases} \tag{29}$$

This form motivates us that the optimal momentum coefficient should be constants in two modes. When $u(t)$ has values between $[0, \infty)$, the optimal control becomes a restarting scheme. When $u(t) = 0$, the system moves fast without any damping. When $u(t) = \infty$, the system loses velocity instantly, and restarts from where it stopped.

3.3 Energy function approach: Why is gradient restarting useful?

When $u(t) = 0$, the dynamic system is given as

$$\ddot{X} = -\nabla f(X) \tag{30}$$

We can easily see that the system is hamiltonian, and the function

$$E(X) = f(X) + \frac{1}{2}|\dot{X}|^2 \tag{31}$$

is conserved, by taking derivative of t. In this setting, moving along with no damping conserves energy, and restarting instantly removes the kinetic energy of the particle. Thus, it is intuitive that restarting at the position of largest velocity is good, because it will remove the most energy. As $E(X)$ is preserved, the position with minimum $f(X)$ will have largest kinetic energy. As gradient descent restarts whenever it is a local minimum of $f(X(t))$, it restarts when it has locally maximal kinetic energy, which makes it a useful strategy. This analysis shows that for this dynamics, the gradient restarting scheme is equivalent to the speed restarting scheme presented in [1], where the algorithm restarts whenever the norm of $\dot{X}$ decreases.

3.4 Linear convergence in quadratic case

We show that the system $X'' = -\nabla f(X)$ with gradient restarting (restarting whenever $\langle \nabla f(X), X' \rangle > 0$) converges linearly for strongly convex quadratic function $f = \frac{1}{2}\langle X, \Lambda X \rangle$ when Λ is a diagonal matrix with positive different diagonal entries and $X \in \mathbb{R}^d$. First, we show that for each restarting, the function decreases by a certain global ratio.

Theorem 1. *For initial x_0, suppose the system restarts after time T. There exists a constant $C < 1$ that is independent of x_0, that satisfies*

$$\frac{f(X(T))}{f(X(0))} \leq C$$

Proof. The solution of the ODE is given as

$$X_i(t) = x_{0,i} cos(\lambda_i t) \tag{32}$$

for $i = 1, 2, ..., d$. Without loss of generality, assume $0 < \lambda_1 \leq \lambda_2 ... \leq \lambda_n$. Then,

$$\frac{df(X)}{dt} = \sum_{i=1}^{n} -\lambda_i^2 x_{0,i}^2 cos(\lambda_i t) sin(\lambda_i t) \tag{33}$$

holds. If $t \leq \frac{\pi}{2\lambda_n}$, as $\cos(\lambda_i t)\sin(\lambda_i t) > 0$, we can see that $\frac{df(X)}{dt}(t) < 0$ holds, and

$$\frac{df(X)}{dt}(T) = 0, \frac{\pi}{2\lambda_n} \leq T \tag{34}$$

also holds.
We know that $f(X(t))$ is a decreasing function when $t \leq T$. This leads to

$$\frac{f(X(T))}{f(X(0))} \leq \frac{f(X(\frac{\pi}{2\lambda_n})}{f(X(0))} \leq \frac{\sum_{i=1}^{n} \lambda_i^2 x_{0,i}^2 \cos(\frac{\lambda_i \pi}{2\lambda_n})^2}{\sum_{i=1}^{n} \lambda_i^2 x_{0,i}^2} \leq \max_{i=1,2,\ldots,n} \cos(\frac{\lambda_i \pi}{2\lambda_n})^2 < 1 \tag{35}$$

And taking $C = \max_{i=1,2,\ldots,n} \cos(\frac{\lambda_i \pi}{2\lambda_n})^2$ finishes the proof.

□

Also, we can find an upper bound of restarting time T independent with the starting position. The proof starts with Minkowski's theorem, which illustrates a condition for a convex set S in $\mathbb{R}^d$ to have an integer point in it.

Lemma 2 (Minkowski)**.** *A convex set S that is symmetric with respect to the origin contains a non-zero integer point if $vol(S) > 2^d$*

Lemma 3. *For any $\lambda_1, \lambda_2, ..., \lambda_n > 0$ and $\delta > 0$, there exists a real number x where the distance between $\lambda_i x$ and the closest integer to it has distance smaller than δ for all i = 1, 2, ..., d.*

Proof. Consider the convex set

$$S(r) = \cup_{-r \leq x \leq r} C(\lambda x, \delta)$$

that is symmetric with respect to the origin. (C(a,b) is a n-dimensional sphere with center a and radius b, $\lambda = [\lambda_1, \lambda_2, ..., \lambda_n]$). The volume of $S(r)$ diverges as r approaches infinity. Thus, when we take r large enough so that $S(r) > 2^n$, we can find x between 0 and r where the distance between $\lambda_i x$ and the closest integer has distance smaller than δ for all i = 1, 2, ..., d.

□

Theorem 2. *For initial x_0, suppose the system restarts after time T. There exists an upper bound of T which is given as a function of Λ.*

Proof. For C in theorem (1), let $\epsilon = \frac{1-C}{2}$. Also, let $\delta < \frac{\pi}{2}$ be a number that satisfies

$$|x| < \delta \leftrightarrow |\sin^2(x)| < \epsilon \tag{36}$$

From Lemma (3), we can find K where $\lambda_1 K$, $\lambda_2 K$, ... , $\lambda_n K$ and the corresponding closest integers for each are all closer than $\frac{\delta}{\pi}$. For that K,

$$|\sin^2(\lambda_i \pi K)| < \epsilon \tag{37}$$

holds, and

$$1 - \epsilon < \cos^2(\lambda_i \pi K) \tag{38}$$

also holds. Thus,

$$f(X(T)) < Cf(X(0)) < f(X(0))(1-\epsilon) < f(X(\pi K)) \tag{39}$$

holds. If $\pi K < T$, as $f(X(t))$ is a decreasing function for $0 \leq t \leq T$, we can obtain contradiction because $f(X(\pi K)) > f(X(T))$. Thus, $T \leq \pi K$ holds, and we can obtain an upper bound πK of restarting time T that is only a function of Λ.

□

Combining theorems (1) and (2), we get the linear convergence result immediately:

Corollary 1. *$f(X(t)) \leq f(X(0))C^{\frac{t}{T_0}}$, where T_0 is the upper bound of restarting time T given in theorem (2)*

3.5 Analysis in general case

For the general case, suppose f is a strongly convex function with strong convexity constant μ, and has Lipschitz continuous gradient with Lipschitz constant L. Let's consider the system $X'' = -\nabla f(X)$ with gradient restarting. It is possible to show that the function value decreases with a specific ratio for each restarting. However, it is a hard question whether there exists a global upper bound for the restarting time: We couldn't figure out whether the restarting happens in finite time or not.

Theorem 3. *For initial $X(0)$, suppose the system restarts after time T. Then,*

$$\frac{f(X(T)) - f^*}{f(X(0)) - f^*} \leq \frac{4L}{4L + \mu}$$

holds.

Proof. Let's denote

$$F(t) = \langle \nabla f(X), \dot{X} \rangle \tag{40}$$

Initially, $\dot{X}(0) = 0$, thus $F(0) = 0$ holds. Also, when restarting happens, $F(T) = 0$ should hold. Thus, there exists t_0 such that $F'(t_0) = 0$ in $(0, T)$. The derivative of F is given as

$$F'(t) = \langle D^2 f(X)\dot{X}, \dot{X} \rangle + \langle \nabla f(X), \ddot{X} \rangle \tag{41}$$

As $F'(t_0) = 0$ and $\ddot{X} = -\nabla f(X)$ for t in $(0, T)$, we can conclude that

$$\langle \nabla f(X(t_0)), \nabla f(X(t_0)) \rangle = \langle D^2 f(X(t_0))\dot{X}(t_0), \dot{X}(t_0) \rangle \tag{42}$$

holds. We know that ∇f is Lipschitz continuous with constant L, meaning $D^2 f(X(t_0)) \preceq LI$, and

$$\|\nabla f(X(t_0))\|^2 \leq L \left\| \dot{X}(t_0) \right\|^2 \tag{43}$$

must hold. By convexity of f we know that

$$f(X(t_0)) - f^* \leq \langle \nabla f(X(t_0)), X(t_0) - X^* \rangle \leq \|\nabla f(X(t_0))\| \, \|X(t_0) - X^*\| \tag{44}$$

and

$$\|\nabla f(X(t_0))\|^2 \geq (\frac{f(X(t_0)) - f^*}{\|X(t_0) - X^*\|})^2 \geq \frac{\mu}{2}(f(X(t_0)) - f^*) \tag{45}$$

must hold. The last inequality follows from the strong convexity of f. Also, we know that the quantity $f(X) + \frac{1}{2}\left\|\dot{X}\right\|^2$ is preserved when t is in $(0, T)$. Thus,

$$\left\|\dot{X}(t_0)\right\|^2 = 2(f(X(0)) - f(X(t_0))) \tag{46}$$

must hold. Plugging the results of (45) and (46) in (43), we get

$$\mu(f(X(0)) - f^*) \leq 4L(f(X(0)) - f(X(t_0))) \tag{47}$$

Simplifying the inequality leads to

$$\frac{f(X(t_0)) - f^*}{f(X(0)) - f^*} \leq \frac{4L}{4L + \mu} \tag{48}$$

As $f(X(T)) \leq f(X(t_0))$, we obtain the wanted inequality.

□

4 Discrete Case Analysis

With the motivations from Pontryagin's maximal principle and the intuition that low damping will result in faster convergence, we propose the algorithm below:

Algorithm 2 Proposed algorithm

Require: x_0, $x_{-1} = x_0$, $y_0 = x_0$, s, $\beta \leq 1$

for $i = 0, 1, 2, ...$ **do**
  if $\langle G_s(y_i), x_i - x_{i-1} \rangle > 0$ **then**
    $x_{i+1} = x_i - sG_s(x_i)$
    $y_{i+1} = x_{i+1}$
  else
    $x_{i+1} = y_i - sG_s(y_i)$
    $y_{i+1} = x_{i+1} + (x_{i+1} - x_i)$
  end if
end for

Here, f, the function that we want to optimize, can be written as a sum of a smooth function g and a non-differentiable function h, and $G_s(x)$ is the proximal gradient of f. Thus, algorithm can be thought of as a "Proximal Gradient" version of the algorithm proposed in [3], with the use of Nesterov momentum update rule. The overall proof is similar, but it is meaningful that we actually show that the proof can be extended in the case of proximal gradients, and for Nesterov updates.

Assuming g is strongly convex with constant μ and has Lipschitz continuous gradient with constant

L, we can prove linear convergence of the algorithm.

First, we will often use the inequality that involves proximal gradients. Let's call it the fundamental inequality of proximal gradient.

Lemma 4 (Fundamental inequality of proximal gradient)**.** *For* $0 \leq s \leq \frac{1}{L}$,

$$f(x - sG_s(x)) \leq f(z) + \langle G_s(x), x - z\rangle - \frac{s}{2}\|G_s(x)\|^2 - \frac{\mu}{2}\|x - z\|^2 \tag{49}$$

holds for all $x, z \in \mathbb{R}^d$

We assume that $0 \leq s \leq \frac{1}{L}$ holds for s that is used below.

Now, like theorem 3, we can show that when $\langle G_s(y_k), x_k - x_{k-1}\rangle \leq 0$, there is a global constant $C < 1$ that satisfies

$$\frac{f(x_{k+1}) - f^*}{f(x_k) - f^*} \leq C$$

. Actually, we prove a stronger result, where C decreases as k increases. We begin with a simple lemma:

Lemma 5. *For any* $x \in \mathbb{R}^d$,

$$\frac{2\mu}{1 - \mu s}(f(x - sG_s(x)) - f^*) \leq \|G_s(x)\|^2 \tag{50}$$

holds.

Proof. Plug $z = x^*$ in lemma 4. We get

$$f(x - sG_s(x)) \leq f^* + \langle G_s(x), x - x^*\rangle - \frac{s}{2}\|G_s(x)\|^2 - \frac{\mu}{2}\|x - x^*\|^2 \tag{51}$$

Simplify to get

$$\begin{aligned}
f(x - sG_s(x)) - f^* &\leq \langle G_s(x), x - x^*\rangle - \frac{s}{2}\|G_s(x)\|^2 - \frac{\mu}{2}\|x - x^*\|^2 \\
&\leq \|G_s(x)\|\,\|x - x^*\| - \frac{\mu}{2}\|x - x^*\|^2 - \frac{s}{2}\|G_s(x)\|^2 \\
&= \frac{\|G_s(x)\|^2}{2\mu} - (\|x - x^*\|\sqrt{\frac{\mu}{2}} - \frac{\|G_s(x)\|}{\sqrt{2\mu}})^2 - \frac{s}{2}\|G_s(x)\|^2 \\
&\leq \frac{\|G_s(x)\|^2}{2\mu} - \frac{s}{2}\|G_s(x)\|^2 \\
&= \frac{1 - \mu s}{2\mu}\|G_s(x)\|^2
\end{aligned}$$

Multiply $\frac{2\mu}{1-\mu s}$ on both sides to get the desired result.

□

Theorem 4. *Suppose restarting has not occured from* x_0 *to* x_k*, i.e.* $\langle G_s(y_i), x_i - x_{i-1}\rangle \leq 0$ *holds for* $i = 1, 2, \ldots, k$*. Then,*

$$f(x_i) - f^* \leq (1 - \mu s - (i-1)\mu^2 s^2)(f(x_{i-1}) - f^*)$$

and

$$f(x_k) - f^* \leq \prod_{i=0}^{k-1}(1 - \mu s - i\mu^2 s^2)(f(x_0) - f^*)$$

holds, assuming monotonic decrease for $f(x_k)$*.*

Proof. Plug in $x = y_k, z = x_k$ to lemma 4 to get

$$f(x_{k+1}) = f(y_k - sG_s(y_k)) \leq f(x_k) + \langle G_s(y_k), y_k - x_k\rangle - \frac{s}{2}\|G_s(y_k)\|^2 - \frac{\mu}{2}\|y_k - x_k\|^2 \tag{52}$$

for any $k \in \mathbb{N}$. As $y_i - x_i = x_i - x_{i-1}$ for $i = 1, 2, ..., k$, we get

$$\begin{aligned} f(x_i) &\leq f(x_{i-1}) + \langle G_s(y_{i-1}), y_{i-1} - x_{i-1}\rangle - \frac{s}{2}\|G_s(y_{i-1})\|^2 - \frac{\mu}{2}\|y_{i-1} - x_{i-1}\|^2 \\ &= f(x_{i-1}) + \langle G_s(y_{i-1}), x_{i-1} - x_{i-2}\rangle - \frac{s}{2}\|G_s(y_{i-1})\|^2 - \frac{\mu}{2}\|x_{i-1} - x_{i-2}\|^2 \\ &\leq f(x_{i-1}) - \frac{\mu}{2}\|x_{i-1} - x_{i-2}\|^2 - \frac{s}{2}\|G_s(y_{i-1})\|^2 \end{aligned}$$

Moreover, we can see that

$$\begin{aligned} \|x_k - x_{k-1}\|^2 &= \|-sG_s(y_{k-1}) + y_{k-1} - x_{k-1}\|^2 \\ &= s^2\|G_s(y_{k-1})\|^2 - 2s\langle G_s(y_{k-1}), y_{k-1} - x_{k-1}\rangle + \|y_{k-1} - x_{k-1}\|^2 \\ &\geq s^2\|G_s(y_{k-1})\|^2 + \|y_{k-1} - x_{k-1}\|^2 \\ &= s^2\|G_s(y_{k-1})\|^2 + \|x_{k-1} - x_{k-2}\|^2 \end{aligned}$$

holds. The inequality follows from the assumption that restarting did not happen at step k. If we repeat, we can obtain the inequality

$$\|x_k - x_{k-1}\|^2 \geq \sum_{i=0}^{k-1} s^2\|G_s(y_i)\|^2 \tag{53}$$

Use lemma 5 to obtain

$$\begin{aligned} \|x_{k-1} - x_{k-2}\|^2 &\geq \sum_{i=0}^{k-2} s^2\|G_s(y_i)\|^2 \\ &\geq \frac{2\mu s^2}{1 - \mu s}(\sum_{i=1}^{k-1}(f(x_i) - f^*)) \end{aligned}$$

From the fact that $f(x_k) \leq f(x_{k-1}) - \frac{\mu}{2}\|x_{k-1} - x_{k-2}\|^2 - \frac{s}{2}\|G_s(y_{k-1})\|^2$, we get

$$f(x_k) \leq f(x_{k-1}) - \frac{\mu^2 s^2}{1 - \mu s}(\sum_{i=1}^{k-1}(f(x_i) - f^*)) - \frac{\mu s}{1 - \mu s}(f(x_k) - f^*) \tag{54}$$

Subtract f^* on both sides and simplify to get

$$f(x_k) - f^* \leq (1 - \mu s)(f(x_{k-1}) - f^*) - \mu^2 s^2 (\sum_{i=1}^{k-1} (f(x_i) - f^*)) \tag{55}$$

Assuming monotonicity, we get

$$f(x_k) - f^* \leq (1 - \mu s)(f(x_{k-1}) - f^*) - (k-1)\mu^2 s^2 (f(x_{k-1}) - f^*)) \tag{56}$$

and

$$f(x_k) - f^* \leq (1 - \mu s - (k-1)\mu^2 s^2)(f(x_{k-1}) - f^*)$$

holds. As the inequality holds for k that restarting has never occured since x_0, the inequality holds for all $i \leq k$. Multiplying the inequalities leads to

$$f(x_k) - f^* \leq \prod_{i=0}^{k-1} (1 - \mu s - i\mu^2 s^2)(f(x_0) - f^*) \tag{57}$$

□

Remark: The bound in theorem 4 is not tight. Actually, when we denote A_k to be a sequence that satisfies

$$f(x_k) - f^* \leq A_k (f(x_0) - f^*) \tag{58}$$

A_k satisfies

$$A_k = (1 - \mu s - \mu^2 s^2) A_{k-1} - \mu^2 s^2 \sum_{i=1}^{k-2} \frac{1}{A_i} \tag{59}$$

which is a direct consequence of inequality 55

A direct corollary of theorem 4 is that there exists an upper bound of the restarting step N.

Corollary 2. *The restarting step N cannot exceed $\frac{1-\mu s}{\mu^2 s^2}$*

Proof. We know that from theorem 4,

$$0 \leq f(x_k) - f^* \leq \prod_{i=0}^{k-1} (1 - \mu s - i\mu^2 s^2)(f(x_0) - f^*) \tag{60}$$

must hold. Thus, the restarting step N should satisfy

$$0 \leq 1 - \mu s - N\mu^2 s^2 \tag{61}$$

leading to the claimed result. □

Now, we discuss what happens when restarting occurs.

Theorem 5. *Suppose $\langle G_s(y_k), x_k - x_{k-1} \rangle > 0$, thus restarting occured. Then,*

$$f(x_{k+1}) - f^* \leq (1 - \mu s)(f(x_k) - f^*)$$

holds.

Proof. Plug in $x = z = x_k$ at lemma 4. As $x_{k+1} = x_k - sG_s(x_k)$ holds,

$$f(x_{k+1}) \leq f(x_k) - \frac{s}{2} \|G_s(x_k)\|^2 \tag{62}$$

also holds.
Now, from lemma 5, we know that

$$\frac{2\mu}{1-\mu s}(f(x_{k+1}) - f^*) \leq \|G_s(x_k)\|^2 \tag{63}$$

holds. Using this, we can obtain

$$f(x_{k+1}) \leq f(x_k) - \frac{\mu s}{1-\mu s}(f(x_{k+1}) - f^*) \tag{64}$$

Subtracting f^* on both sides and simplify to get the wanted result.

□

In both regime, when the algorithm restarts or does not restart, $f(x_k) - f^*$ decreases with a certain ratio, meaning that f converges linearly to f^*. When we denote the number of restarts as M during N steps of the algorithm, and denote $r_1, r_2, ..., r_M$ as the number of steps before each restart, we get the following convergence rate.

Corollary 3. *For the notations above,*

$$f(x_N) - f^* \leq (1-\mu s)^M \prod_{t=1}^{M} \prod_{i=0}^{r_t - 1} (1 - \mu s - i\mu^2 s^2)(f(x_0) - f^*)$$

Note that for vanila gradient descent, the decrease ratio is $1 - \mu s$. This means that the proposed algorithm has better convergence rate than vanila proxmial gradient descent. However, the bound is not very tight: as μs is often a very small number $\mu^2 s^2$ is also very small, and the term $i\mu^2 s^2$ is often negligible. Researches on better bound on the convergence rate of the algorithm might be meaningful.

5 Numerical Experiments

I experimented different restarting schemes on different optimization tasks to compare each scheme and show that the presented method can be beneficial in most cases. Vanila Nesterov algorithm, gradient restarting scheme, speed restarting scheme, and the proposed algorithm were experimented. Some objective functions in the experiment were not strongly convex, or even convex.

Square LASSO. $f(x) = \frac{1}{2}| \|Ax - b\|^2 + \lambda |x|_1$, where $|\cdot|_1$ denotes the l1 norm. A is a 500×500 matrix x, b is a vector of size 500, regularization parameter $\lambda = 4$, and the learning rate is set to be 5e-4. Each entry of A is initialized to follow unit normal distribution, each entry in b is sampled from $N(0, 9)$.

Sparse LASSO. $f(x)$ is the same as above, but A is a 5000×50000 matrix, x, b is a vector of size 50000, $\lambda = 5$, and the learning rate is set to be 1e-2. Each entry of A has probability 0.5% of

being nonzero, and each entry is sampled from the normal distribution of zero mean and variance 0.04. b is initialized to $Ax_0 + z$, where x_0 and z has random normal entries.

Square NNLS. $f(x) = \frac{1}{2}||Ax - b||^2$, and the function is optimized under the constraint $x \geq 0$. A is a 500×500 matrix, x, b is a vector of size 500, and the learning rate is set to be 5e-4. Entries of A is sampled from unit normal distribution and entries of b is sampled from $N(0, 9)$

Log-Sum-Exp. $f(x) = \rho \log[\sum_{i=1}^{n} \exp((\frac{\langle a_i, x\rangle - b_i)}{\rho})]$, where $n = 50$, a_i, x are vectors of size $m = 200$, b is a vector of size n, $\rho = 20$ and learning rate is set to be 0.5.

Logistic. $f(x) = \sum_{i=1}^{n} -y_i a_i^T x_i + \log(1 + e^{a_i^T x_i})$, each a_i's and x_i are vectors of size m, and y is a vector of size n. n was set to 500, m was set to 100, and the learning rate was 1e-3. In the experiment x_i and each a_i's entries were sampled from unit normal distribution, and each entry of y was 0 or 1 sampled from a logistic distribution.

Logistic+l1 regularization. $f(x) = \sum_{i=1}^{n} -y_i a_i^T x_i + \log(1 + e^{a_i^T x_i}) + \lambda |x|_1$. The setting was identical with the vanila logistic case, except $\lambda = 1$.

The experiment result for different experiments can be seen in figure 3. From the result, we can see that the proposed algorithm converges faster than other restarting-based algorithms. The most effective task to use the proposed algorithm was Sparse Lasso: in contrast to other tasks, gradient restarting fails to restart early, resulting in a slow convergence. On the other hand, speed restarting and the proposed algorithm restarts much earlier than gradient restarting, thus showing faster convergence before gradient restarting restarts.

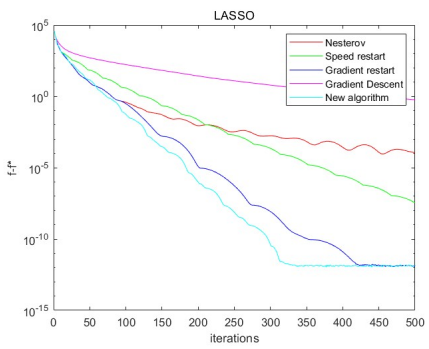
Moreover, the proposed algorithm shows a more stable curve in the decrease of function value. From figure 3 we can see that speed restarting and proposed algorithm shows less ripples, where gradient restart shows more ripples. From [1] the fact that speed restarting shows a more stable curve in the change of function value was reported: as the purposed algorithm's continuous counterpart can also be understood as speed restarting scheme, (see 3.3) the proposed algorithm acts similar like a speed restarting scheme, thus being stable.

To summarize, the purposed algorithm is both fast and stable in the change of function value, where it uses the best of both gradient and speed restarting.

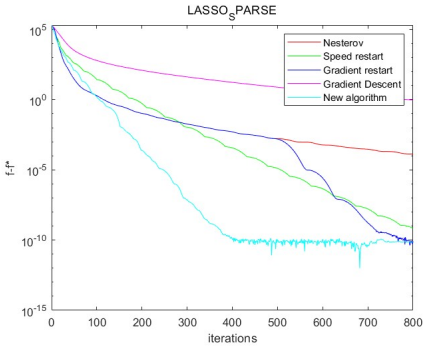
We also present a table that shows how many steps are needed to reach $|f - f^*| \leq \epsilon$, where $\epsilon = 1e - 10$. 100 different experiments were done and were averaged. From the table we can clearly see the improvement in convergence.

	Nesterov	SpeedRestart	GradRestart	Presented
Square LASSO	2223.9	616.9	359.5	**282.2**
Sparse LASSO	>10000	926.2	694.2	**407.9**
Square NNLS	496.8	206.4	129.6	**107.5**
Log-Sum-Exp	8426.2	1481.4	763.9	**569.9**
Logistic	699.2	369.6	217.4	**170.5**
Logistic-l1reg	721.1	291.0	171.6	**139.9**

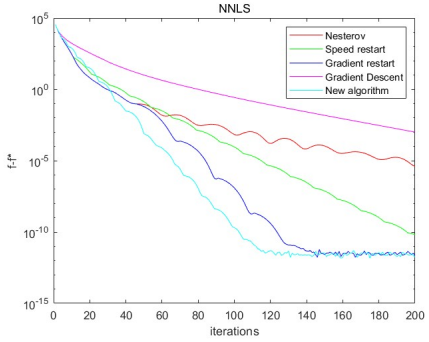
Table 1: Steps needed to approach $|f - f^*| \leq \epsilon$



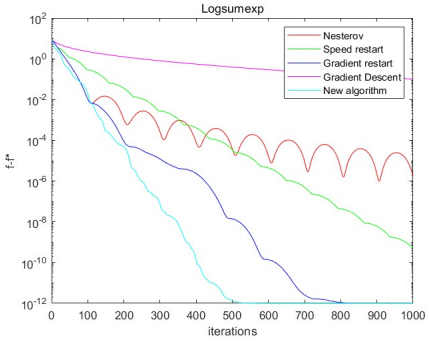
(a) Square Lasso



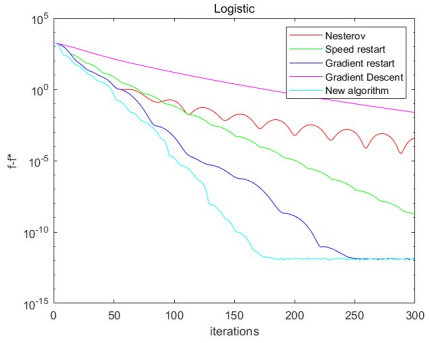
(b) Sparse Lasso



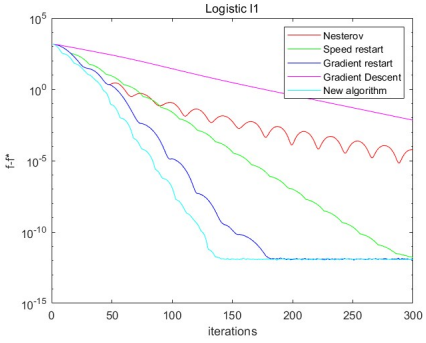
(c) Square NNLS



(d) Log-sum-exp



(e) Logistic



(f) Logistic + l1 regularization

Figure 3: Numerical Experiments on various tasks

6 Conclusion

For the graduate project I aimed find design a damping coefficient that can speed up convergence of the accelerated gradient descent algorithm, and discuss its optimality. Though I couldn't find the "optimal" damping, I could present a dynamics and its corresponding optimization algorithm that converges faster than restarting methods of original Nesterov algorithm. Also, from Pontryagin's maximal principle I could see that the optimal control to make the system converge fastest must be of the form of restarting. Last, I extended the proof of [3] to proximal gradients and showed explicit proof in the Nesterov iteration case.

However, the work is incomplete in two ways: First, in continuous case, it is still unknown whether the system restarts in finite time. Second, the convergence rate obtained in Corollary 3 is weak. We expect further research to fill these gaps, and thoroughly explain why gradient restarting works so well.

References

[1] Weijie Su, Stephen Boyd, and Emmanuel Candes. "A differential equation for modeling Nesterov's accelerated gradient method: Theory and insights". In: *Advances in neural information processing systems* 27 (2014), pp. 2510–2518.

[2] Brendan O'donoghue and Emmanuel Candes. "Adaptive restart for accelerated gradient schemes". In: *Foundations of computational mathematics* 15.3 (2015), pp. 715–732.

[3] Can Kizilkale. "Topics in Optimization: Restarted Moment Based Methods, Underdetermined Systems and Trade Networks". PhD thesis. UC Santa Barbara, 2019.

[4] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*. Vol. 87. Springer Science & Business Media, 2003.