SIAM J. SCI. COMPUT.
Vol. 43, No. 5, pp. A3184–A3211

SEARCH DIRECTION CORRECTION WITH NORMALIZED GRADIENT MAKES FIRST-ORDER METHODS FASTER*

YIFEI WANG†, ZEYU JIA‡, AND ZAIWEN WEN§

Abstract. The so-called fast inertial relaxation engine is a first-order method for unconstrained smooth optimization problems. It updates the search direction by a linear combination of the past search direction, the current gradient, and the normalized gradient direction. We explore more general combination rules and call this generalized technique the search direction correction (SDC). SDC is extended to composite and stochastic optimization problems as well. Deriving from a second-order ODE, we propose a fast inertial search direction correction (FISC) algorithm as an example of methods with SDC. We prove the $\mathcal{O}(k^{-2})$ convergence rate of FISC for convex optimization problems. Numerical results on sparse optimization, logistic regression, as well as deep learning demonstrate that our proposed methods are quite competitive to other state-of-the-art first-order algorithms.

Key words. first-order methods, search direction correction, Lyapunov function, composite optimization, stochastic optimization

AMS subject classifications. 49M07, 65C60, 65K05, 90C06

DOI. 10.1137/20M1335480

1. Introduction.

We take the following optimization problem into consideration:

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}) = \psi(\mathbf{x}) + h(\mathbf{x}), \tag{1.1}$$

where ψ is a smooth function and h is a possibly nonsmooth convex function. In machine learning, ψ often has the form

$$\psi(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} \psi_i(\mathbf{x}), \tag{1.2}$$

where ψ_i is the prediction error to the ith sample. Since the dimension of the variable x and the number of samples N are often extremely huge, first-order and/or stochastic algorithms are frequently used for solving (1.1).

First-order algorithms only use the information of the function value and the gradient. The vanilla gradient descent method is the simplest algorithm with convergence guarantees. Adding momentum to the current gradient has been an efficient technique to accelerate the convergence. This type of algorithms includes the Nesterov accelerated method [20], the Polyak heavy-ball method [23], and the nonlinear conjugate gradient method [6]. Except the last one, these methods can be extended

*Submitted to the journal's Methods and Algorithms for Scientific Computing section May 1, 2020; accepted for publication (in revised form) April 23, 2021; published electronically September 16, 2021.

https://doi.org/10.1137/20M1335480

Funding: The work of the third author was partially supported by the National Key R&D Program of China (2018YFC0704305), by NSFC grant 11831002, and by Beijing Academy of Articial Intelligence.

†Department of Electrical Engineering, Stanford University, Stanford, CA 94305-9505 USA (wangyf18@stanford.edu).

‡Department of EECS, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (zyjia@mit.edu).

§Corresponding author. Beijing International Center for Mathematical Research and Center for Data Science, Peking University, China (wenzw@pku.edu.cn).

to cases where h is nonsmooth, by replacing the gradient with the so-called proximal gradient. Meanwhile, it is proved in [21] that first-order algorithms cannot achieve a convergence rate better than $\mathcal{O}(1/k^2)$. In this way, the convergence rate of the Nesterov accelerated method matches this lower bound exactly.

Lately, a new technique borrowed from ODE and dynamical system has been used to analyze the behavior of optimization algorithms, especially first-order methods. Several ODEs corresponding to different types of Nesterov accelerated methods when the step size converges to zero are analyzed in [25]. With specifically designed Lyapunov functions, they obtained a proportional convergence rate for these ODEs and for Nesterov accelerated methods. Wibisono, Wilson, and Jordan [27] and [28] generalized this technique to a broader class of first-order algorithms. Zhang et al. [33] proposed a different type of Lyapunov function and obtained a convergence competitive to Nesterov accelerated methods. Attouch et al. [1] analyze the ODE of such an inertial system with extra Hessian driven damping and propose corresponding discrete-time algorithms.

To deal with the large sample numbers N in evaluating full gradients, stochatic methods are introduced. The stochastic gradient descent method (SGD) is the stochastic version of the vanilla gradient descent method. However, SGD may suffer from the large variance of stochastic gradients during its iterations. To tackle this problem, Stochastic Variance Reduced Gradient (SVRG) [12], Stochastic Average Gradient (SAG) [24], and SAGA [7] introduce variance reduction techniques and achieve acceleration compared to SGD.

A natural question arises: how do we design efficient first-order stochastic methods to improve the optimization procedure? An optimization algorithm called the fast inertial relaxation engine (FIRE) [4] from molecular dynamics seems to be quite promising. FIRE is proposed for finding the atomic structures with the minimum potential energy. Involving an extra term of the velocity correction along the gradient direction with the same magnitude of the current velocity, and adopting a carefully designed restarting condition, FIRE can practically achieve better performance than other first-order methods including the conjugate gradient method. It is even competitive to the limited-memory BFGS [16] in several test cases. However, neither is the choice of molecular dynamics integrator specified nor the convergence rate given in [4].

Motivated by first-order algorithms and FIRE, we introduce a family of first-order methods with the search direction correction (SDC) and propose the fast inertial search direction correction (FISC) algorithm. Our contributions are listed as follows:

- We adapt FIRE in molecular dynamics to solve general smooth and nonsmooth optimization problems. We explore more general combination rules of updating search direction in FIRE and generalize it into a framework of first-order methods with SDC. We allow more choices for step sizes, such as applying a line search technique to find a step size satisfying the Armijo conditions or the nonmonotone Armijo conditions. The basic restarting condition ensures the global convergence for methods with SDC. Furthermore, SDC is extended to composite optimization and stochastic optimization problems.
- Second-order ODEs of methods with SDC in continuous time are derived via taking the step size to zero. By constructing a Lyapunov function and analyzing its derivative, we prove that the ODE corresponding to FISC has the convergence rate of $O(1/t^2)$ on smooth convex optimization problems. We also build a discrete Lyapunov function for FISC in the discrete case. On composite optimization problems, FISC is proven to have the $\mathcal{O}(1/k^2)$ convergence rate.

- Our algorithms are tested on sparse optimization, logistic regression, and deep learning. Numerical experiments indicate that our algorithms are quite competitive to other state-of-the-art first-order algorithms.

1.1. Organization. This paper is organized as follows. We present the update rule of methods with SDC including FISC in section 2. In section 3, the ODE perspective of FISC is used to provide a necessary condition for the convergence. The global convergence of methods with SDC and the convergence rate of FISC are discussed in section 4. Finally, in section 5, we present numerical experiments to compare FISC, FIRE, and other first-order algorithms.

1.2. Preliminaries. We use standard notation throughout the paper. $\|\cdot\|$ is the standard Euclidean norm and $\langle\cdot,\cdot\rangle$ is the standard Euclidean inner product. $\mathcal{F}_L$ stands for the class of convex and differentiable functions with L-Lipschitz continuous gradients. $\mathcal{F}$ represents the class of convex and differentiable functions. $\mathbb{R}^+$ is the collection of nonnegative real number. $[N]$ denotes $\{1, 2, \ldots, N\}$.

2. The framework of SDC. In this section, we introduce the framework of first-order methods with SDC to solve smooth optimization problems (1.1) with $h = 0$. Based on this framework, we extend SDC to composite optimization problems, stochastic optimization problems, and deep learning. We also compare SDC with other first-order methods with momentum terms.

2.1. A family of first-order methods with SDC. In this subsection, we focus on solving smooth optimization problems (1.1) with $h = 0$. It involves two sequences of parameters $\{\beta_k\}_{k=1}$ and $\{\gamma_k\}_{k=1}$ and introduces a velocity $\mathbf{u}$ as a search direction to update $\mathbf{x}$.

We start with an initial guess $\mathbf{x}_0$ and an initial velocity $\mathbf{u}_0 = 0$. In the beginning of the $(k+1)$th iteration, we determine whether $\mathbf{u}_k$ is a descent direction by introducing a restarting condition

$$\varphi_k = \langle -\nabla f(\mathbf{x}_k), \mathbf{u}_k \rangle \geq 0. \tag{2.1}$$

When this condition is satisfied, we update

$$\mathbf{u}_{k+1} = (1-\beta_k)\mathbf{u}_k - \gamma_k \frac{\|\mathbf{u}_k\|}{\|\nabla f(\mathbf{x}_k)\|}\nabla f(\mathbf{x}_k) - \nabla f(\mathbf{x}_k). \tag{2.2}$$

When $k = 0$, we directly have $\mathbf{u}_1 = -\nabla f(\mathbf{x}_0)$ given $\mathbf{u}_0 = 0$, so β_0 and γ_0 need not be specified. We further require β_k and γ_k to satisfy

$$0 \leq \beta_k \leq 1, \quad 0 \leq \gamma_k \leq 1. \tag{2.3}$$

Then we update β_{k+1} and γ_{k+1} as follows.

- In *FIRE* [4], they are updated by
$$\gamma_{k+1} = \beta_{k+1} = d_\beta \beta_k,$$
where $0 < d_\beta < 1$ is a parameter. The initial value of $\{\beta_k\}$ is set to $\beta_1 = 1$, and d_β is given by $d_\beta = 0.99$.
- In *FISC*, β_k and γ_k are parameterized with l_k, i.e.,
$$\beta_k = \frac{r}{l_k - 1 + r}, \quad \gamma_k = \frac{r-3}{l_k - 1 + r}, \tag{2.4}$$
where $r \geq 3$ and $\{l_k\}$ is a sequence of parameters with an initial value of $l_1 = 1$. We update $l_{k+1} = l_k + 1$.

If the condition (2.1) is not met, we restart the system by resetting $\mathbf{u}_{k+1}, \beta_{k+1}$, and γ_{k+1} as

$$\mathbf{u}_{k+1} = -\nabla f(\mathbf{x}_k), \tag{2.5}$$

$$\beta_{k+1} = \beta_1, \gamma_{k+1} = \gamma_1. \tag{2.6}$$

Specifically, in FISC, we reset $l_{k+1} = l_1$.

Then, we calculate the step size s_k. Either of the following choices of s_k is acceptable:

(i) Fix the step size $s_k = s_0$.

(ii) Perform a backtracking line search to find a step size s_k that satisfies the Armijo conditions:

$$f(\mathbf{x}_k - s_k\mathbf{u}_{k+1}) \le f(\mathbf{x}_k) - \sigma s_k \langle \mathbf{u}_{k+1}, \nabla f(\mathbf{x}_k)\rangle, \tag{2.7}$$

where $0 < \sigma < 1$ is a parameter and $s_k = \bar{s}_k\rho^{h_k}$. Here $\bar{s}_k > 0$ is the trial step and h_k is the largest number such that (2.7) holds.

(iii) Perform a nonmonotone line search [32] to find a step size s_k that satisfies nonmonotone Armijo conditions:

$$f(\mathbf{x}_k - s_k\mathbf{u}_{k+1}) \le C_k - \frac{s_k}{2}\langle \mathbf{u}_{k+1}, \nabla f(\mathbf{x}_k)\rangle, \tag{2.8}$$

where $s_k = \bar{s}_k\rho^{h_k}$. Here $\bar{s}_k > 0$ is the trial step and h_k is the largest number such that (2.8) holds. C_k and Q_k are updated as

$$Q_{k+1} = \eta_k Q_k + 1, \quad C_{k+1} = (\eta_k Q_k C_k + f(\mathbf{x}_{k+1}))/Q_{k+1}$$

with initial values $C_0 = f(\mathbf{x}_0), Q_0 = 1$. η_k is selected from $[\eta_{min}, \eta_{max}]$. The existence of s_k is proved in subsection 4.1.

After calculating the step size s_k, we update

$$\mathbf{x}_{k+1} = \mathbf{x}_k + s_k\mathbf{u}_{k+1}. \tag{2.9}$$

Then, we replace k by $k+1$ and check whether convergence criteria are satisfied. A family of first-order methods with SDC is given in Algorithm 2.1.

Algorithm 2.1 A family of first-order methods with SDC.

Input: initial guess $\mathbf{x}_0$, initial value $\mathbf{u}_0 = 0$, other required parameters.

set $k = 0$, fix step size s_0, or calculate it by the line search.
while *Convergence criteria are not met* or $k < N_{max}$ **do**
  Calculate φ_k by (2.1).
  if $\varphi_k \ge 0$ **then**
    Compute $\mathbf{u}_{k+1}$ by (2.2) and update $\beta_{k+1}, \gamma_{k+1}$.
  else
    Set $\mathbf{u}_{k+1}$ by (2.5) and reset $\beta_{k+1}, \gamma_{k+1}$.
  end if
  Fix step size s_k or calculate it using line search techniques.
  Update $\mathbf{x}_{k+1}$ by (2.9), $k \to k+1$.
end while
return $\mathbf{x}_k$

2.2. A variant of FISC. In this subsection, we introduce FISC-nes, a variant of FISC. A detailed derivation of FISC and FISC-nes is shown in section 3. In FISC-nes, $\mathbf{u}_k$ is replaced by an auxiliary variable $\mathbf{y}_k$ and $\{l_k\}_{k=1}$ in FISC-nes remains the same. We start with $\mathbf{x}_0 = \mathbf{x}_{-1}$. Given $\mathbf{x}_k$ and $\mathbf{x}_{k-1}$, the restarting condition uses the quantity

$$\varphi_k = \langle -\nabla f(\mathbf{x}_k), \mathbf{x}_k - \mathbf{x}_{k-1} \rangle .$$

If $\varphi_k \geq 0$, we compute $\mathbf{y}_k$ by

$$\mathbf{y}_k = \mathbf{x}_k + \frac{l_k - 1}{l_k - 1 + r}(\mathbf{x}_k - \mathbf{x}_{k-1}) - \frac{r-3}{l_k - 1 + r} \frac{||\mathbf{x}_k - \mathbf{x}_{k-1}||}{||\nabla f(\mathbf{x}_k)||} \nabla f(\mathbf{x}_k). \tag{2.10}$$

The step s_k is calculated at $\mathbf{y}_k$ using the direction $-\nabla f(\mathbf{y}_k)$. We then update

$$\mathbf{x}_{k+1} = \mathbf{y}_k - s_k \nabla f(\mathbf{y}_k) \tag{2.11}$$

and update l_{k+1}. Otherwise, we calculate the step size s_k at $\mathbf{x}_k$ using the direction $-\nabla f(\mathbf{x}_k)$. Then $\mathbf{x}_{k+1}$ is updated by

$$\mathbf{x}_{k+1} = \mathbf{x}_k - s_k \nabla f(\mathbf{x}_k),$$

and we reset $l_{k+1} = l_1$. We name this algorithm FISC-nes because it has an analogous update rule with Nesterov's accelerated method.

If all restarting conditions are satisfied and the step size is fixed to be s, FISC updates

$$\mathbf{x}_{k+1} = \mathbf{y}_k - s \nabla f(\mathbf{x}_k), \tag{2.12}$$

where $\mathbf{y}_k$ is defined in (2.10), while FISC-nes updates $\mathbf{x}_{k+1}$ via (2.11). In section 4, we prove that with the update rule of FISC-nes (2.11), FISC-nes has an $\mathcal{O}(k^{-2})$ convergence rate. With $r > 3$, FISC-nes has to calculate the gradient twice in updating $\mathbf{x}_{k+1}$, which may be computationally costly. On the other hand, the update rule of FISC (2.12) can be viewed as an approximation of the update rule of FISC-nes (2.11) and it only evaluates the gradient once in each iteration. In short, FISC-nes has better theoretical explanations and the performance of FISC is better in practice.

2.3. SDC for other optimization problems. In this subsection, we present two modifications on the framework of SDC to two important optimization problems in machine learning: composite optimization and stochastic optimization. When the target function $f(\mathbf{x})$ has a nonsmooth regularization term $h(\mathbf{x})$, problem (1.1) turns to a composite optimization problem. A typical example of $h(\mathbf{x})$ is $h(\mathbf{x}) = \|\mathbf{x}\|_1$ in compressed sensing. On the other hand, when the smooth part ψ has the finite sum structure (1.2), problem (1.1) becomes a stochastic optimization problem. This is common in the empirical risk minimization setting. In this case, evaluating the full gradient of ψ is computationally inefficient so we use the stochastic oracle of $\nabla\psi$ instead.

2.3.1. Composite optimization problems. Consider the composite optimization problem (1.1), where $\psi \in \mathcal{F}_L$. Given the convex function h and the step size $s > 0$, we define the proximal mapping of h as

$$\text{prox}_h^s(\mathbf{x}) = \arg\min_{\mathbf{z}} \left(\frac{1}{2s}\|\mathbf{z} - \mathbf{x}\|^2 + h(\mathbf{z}) \right).$$

Based on the proximal mapping, the proximal gradient is defined by

$$G_s(\mathbf{x}) = \frac{\mathbf{x} - \text{prox}_h^s(\mathbf{x} - s\nabla\psi(\mathbf{x}))}{s}.$$

Here we present two ways to modify SDC for composite optimization problems. The first way is to use the proximal gradient. We simply replace the gradient $\nabla f(\mathbf{x})$ in (2.2) by the proximal gradient $G_s(\mathbf{x})$. In the $(k+1)$th iteration, the step size s_k is fixed or calculated at $\mathbf{x}_k$ for the proximal gradient, using line search techniques. The basic restarting condition uses the quantity

$$\varphi_k = \langle \mathbf{u}_k, -G_{s_k}(\mathbf{x}_k) \rangle . \tag{2.13}$$

If $\varphi_k \geq 0$, then we will update $\mathbf{u}_{k+1}$ by

$$\mathbf{u}_{k+1} = (1-\beta_k)\mathbf{u}_k - \gamma_k \frac{\|\mathbf{u}_k\|}{\|G_{s_k}(\mathbf{x}_k)\|} G_{s_k}(\mathbf{x}_k) - G_{s_k}(\mathbf{x}_k). \tag{2.14}$$

Otherwise, $\mathbf{u}_{k+1}$ is reset by

$$\mathbf{u}_{k+1} = -G_{s_k}(\mathbf{x}_k), \tag{2.15}$$

and $\beta_{k+1}, \gamma_{k+1}$ are reset using (2.6). Then $\mathbf{x}_{k+1}$ is calculated by (2.9).

The second way is to use the proximal mapping. We introduce an auxiliary variable $\mathbf{y}_k \in \mathbb{R}^n$ and start with $\mathbf{x}_0 = \mathbf{x}_{-1}$. Given $\mathbf{x}_k$ and $\mathbf{x}_{k-1}$, the restarting condition uses the following quantity:

$$\varphi_k = \langle \mathbf{x}_k - \mathbf{x}_{k-1}, -G_{s_k}(\mathbf{x}_k) \rangle .$$

If $\varphi_k \geq 0$, the step size s_k is fixed or calculated at $\mathbf{x}_k$ for the proximal gradient using similar methods. $\mathbf{y}_k$ is updated by

$$\mathbf{y}_k = \mathbf{x}_k + (1-\beta_k)(\mathbf{x}_k - \mathbf{x}_{k-1}) - \gamma_k \frac{\|\mathbf{x}_k - \mathbf{x}_{k-1}\|}{\|G_{s_k}(\mathbf{x}_k)\|} G_{s_k}(\mathbf{x}_k). \tag{2.16}$$

Then we fix the step size $\bar{s}_k$ or calculate it at $\mathbf{y}_k$ for the proximal mapping, compute

$$\mathbf{x}_{k+1} = \mathbf{y}_k - \bar{s}_k G_{\bar{s}_k}(\mathbf{y}_k), \tag{2.17}$$

and update $\beta_{k+1}, \gamma_{k+1}$. Note that $\mathbf{x}_{k+1}$ is the proximal mapping of $\mathbf{y}_k$, i.e., $\mathbf{x}_{k+1} = \text{prox}_h^{\bar{s}_k}(y_k)$.

Otherwise, we fix s_k or calculate it at $\mathbf{x}_k$ for the proximal mapping, update

$$\mathbf{x}_{k+1} = \mathbf{x}_k - s_k G_{s_k}(\mathbf{x}_k), \tag{2.18}$$

and reset $\beta_{k+1}, \gamma_{k+1}$ by (2.6).

Taking $\beta_k = \frac{r}{l_k - 1 + r}$ and $\gamma_k = \frac{r-3}{l_k - 1 + r}$ in (2.16), we obtain FISC-PM. When the nonsmooth part $h = 0$ in (1.1), FISC-PM reduces to FISC-nes. On the other hand, with $r = 3$ in FISC-PM, FISTA [3] can be recovered.

2.3.2. Stochastic composite optimization problems. Consider the stochastic composite optimization problem (1.1), where ψ has the finite sum form (1.2) and $\psi_i \in \mathcal{F}_L$. In each iteration, we generate stochastic approximations of the gradient via selecting subsamples $\mathcal{T}_k \subset [N]$ uniformly at random. That is, the minibatch stochastic oracle is obtained as follows:

$$\nabla \psi^{(k)}(\mathbf{x}) = \frac{1}{|\mathcal{T}_k|} \sum_{i \in \mathcal{T}_k} \nabla \psi_i(\mathbf{x}). \tag{2.19}$$

Motivated by [30], we also adopt the variance reduced version of stochastic gradient. With an extra parameter $m \in \mathbb{N}$, the stochastic oracle can be as follows:

$$\begin{cases} \text{If } \ k \bmod m = 0, \quad \text{then} \quad \text{set } \tilde{\mathbf{x}} = \mathbf{x}_k \text{ and calculate } \nabla \psi(\tilde{\mathbf{x}}). \\ \text{Compute } \nabla \psi^{(k)}(\mathbf{x}_k) = \dfrac{1}{|\mathcal{T}_k|} \displaystyle\sum_{i \in \mathcal{T}_k} (\nabla \psi_i(\mathbf{x}) - \nabla \psi_i(\tilde{\mathbf{x}})) + \nabla \psi(\tilde{\mathbf{x}}). \end{cases} \tag{2.20}$$

Here k is the current iteration number and m is the number of iterations after which the full gradient $\nabla \psi$ is evaluated at the auxiliary variable $\tilde{\mathbf{x}}$. Similar to [19], this additional noise-free information is stored and utilized in the computation of the stochastic oracles in the following iterations.

Then, the proximal stochastic gradient is calculated by

$$G_{s_k}(\mathbf{x}) = \frac{\mathbf{x} - \operatorname{prox}_h^{s_k}(\mathbf{x} - s_k \nabla \psi^{(k)}(\mathbf{x}))}{s_k}.$$

The condition $\varphi_k \geq 0$ is evaluated using (2.13). If it is satisfied, we update the velocity $\mathbf{u}_{k+1}$ by (2.14). Otherwise, we reset $\mathbf{u}_{k+1}$, $\beta_{k+1}, \gamma_{k+1}$ by (2.15) and (2.6). SDC for the stochastic optimization can be obtained by setting the nonsmooth part $h = 0$ in (1.1).

2.4. SDC in deep learning. We also adapt SDC to the deep learning setting. In deep learning, h is usually the squared ℓ_2 norm of $\mathbf{x}$ and each ψ_i in (1.1) can be highly nonconvex. The function f can be nonsmooth for neural networks with nonsmooth activations. We make the following changes in updating rules. In the $(k+1)$th iteration, we first calculate the "momentum and gradient update" on $\mathbf{u}_k$ as follows:

$$\tilde{\mathbf{u}}_k = \alpha \mathbf{u}_k - \mathbf{g}_k,$$

where $0 < \alpha < 1$ is a parameter and $\mathbf{g}_k$ is the stochastic subgradient of f evaluated at $\mathbf{x}_k$ through back-propagation. The basic restarting condition becomes

$$\varphi_k = \langle \tilde{\mathbf{u}}_k, -\mathbf{g}_k \rangle .$$

If $\varphi_k \geq 0$, we calculate $\mathbf{u}_{k+1}$ by correcting $\tilde{\mathbf{u}}_k$ to

$$\mathbf{u}_{k+1} = (1 - \beta_k)\tilde{\mathbf{u}}_k - \gamma_k \frac{\|\tilde{\mathbf{u}}_k\|}{\|\mathbf{g}_k\|} \mathbf{g}_k. \tag{2.21}$$

Otherwise, we set

$$\mathbf{u}_{k+1} = -\mathbf{g}_k.$$

Then, $\mathbf{x}_{k+1}$ is updated by (2.9).

Note that if we simply use $\tilde{\mathbf{u}}_k$ or $-\mathbf{g}_k$ as $\mathbf{u}_{k+1}$, then we will get SGD with momentum or vanilla SGD. The heuristic of the change in the deep learning setting is

that we do not want to restart the algorithm too frequently. (Otherwise, the update in $\mathbf{x}_k$ will become the (stochastic) gradient descent.) We notice that

$$\langle \tilde{\mathbf{u}}_k, -\mathbf{g}_k \rangle = \alpha \langle \mathbf{u}_k, -\mathbf{g}_k \rangle + \|\mathbf{g}_k\|_2^2.$$

Hence, a nonnegative $\langle \mathbf{u}_k, -\mathbf{g}_k \rangle$ implies a nonnegative $\langle \tilde{\mathbf{u}}_k, -\mathbf{g}_k \rangle$. With this modification, the algorithm may restart less frequently.

2.5. The comparison with other first-order methods. In this subsection, we compare first-order methods with SDC with the Nesterov's accelerated method with restarting [22], the heavy-ball method [23], and the nonlinear conjugate gradient (CG) method [6].

The Nesterov's accelerated method with restarting. Suppose that the step size is fixed, i.e., $s_k = s$. Taking the limiting process $s \to 0$, the restarting condition (2.1) essentially keeps $\langle \dot{\mathbf{x}}, \nabla f(\mathbf{x}) \rangle$ negative. This coincides with the heuristic in [22], where they proposed a procedure termed as gradient restarting for the Nesterov's accelerated method. Its update rule is given by

$$\begin{cases} \mathbf{x}_k = \mathbf{y}_{k-1} - s\nabla f(\mathbf{y}_{k-1}), \\ \mathbf{y}_k = \mathbf{x}_k + \dfrac{k-1}{k+2}(\mathbf{x}_k - \mathbf{x}_{k-1}). \end{cases} \tag{2.22}$$

The algorithm restarts with $\mathbf{x}_0 = \mathbf{y}_0 := \mathbf{x}_k$ and resets $k = 0$, whenever

$$\langle \nabla f(\mathbf{y}_k), \mathbf{x}_k - \mathbf{x}_{k-1} \rangle > 0.$$

We shall note that this coincides with FISC-nes when $r = 3$. If one takes step size $s \to 0$, this restarting condition also keeps $\langle \nabla f(\mathbf{x}), \dot{\mathbf{x}} \rangle$ nonpositive along the trajectory and resets k to prevent the coefficient $(k-1)/(k+2)$ from steadily increasing to 1. In numerical experiments, we show that FISC will have better performance with $r > 3$ on several test cases.

The heavy-ball method. Consider the case where no restarting condition is applied and the step size s_k is fixed. The update rule of velocity $\mathbf{u}_{k+1}$ in the heavy-ball method [23] is

$$\mathbf{u}_{k+1} = \beta^{(\mathrm{HB})}\mathbf{u}_k - \nabla f(\mathbf{x}_k). \tag{2.23}$$

Then, the heavy-ball method updates $\mathbf{x}_{k+1}$ in the same way as (2.9). The coefficient of $\mathbf{u}_k$ in the heavy-ball method is a constant $\beta^{(\mathrm{HB})}$, while β_k in FIRE decays exponentially and β_k in FISC decays linearly with regard to k. Compared to the heavy-ball method, FIRE/FISC introduce an extra term $\gamma_k \frac{\|\mathbf{u}_k\|}{\|\nabla f(\mathbf{x}_k)\|}\nabla f(\mathbf{x}_k)$ in updating $\mathbf{u}_{k+1}$. For general convex function f, the Cesáro average of the iterates $\mathbf{x}_k$, $\tilde{x}_k = \frac{1}{k+1}\sum_{i=0}^{k}\mathbf{x}_k$ has $\mathcal{O}(1/k)$ convergence rate in function values (see [9]).

The nonlinear CG method. In this case, we obtain a step size s_k by line search techniques and the update rule of search direction $\mathbf{u}_{k+1}$ reads

$$\mathbf{u}_{k+1} = \beta_k^{(CG)}\mathbf{u}_k - \nabla f(\mathbf{x}_k). \tag{2.24}$$

If $\mathbf{u}_k$ does not have the descent property, i.e., $\langle -\nabla f(\mathbf{x}_k), \mathbf{u}_k \rangle < 0$, CG restarts by setting $\mathbf{u}_{k+1} = -\nabla f(\mathbf{x}_k)$. In FIRE, when $\varphi_k = \langle -\nabla f(\mathbf{x}_k), \mathbf{u}_k \rangle$ in the restarting condition is negative, $\mathbf{u}_{k+1}$ is reset in (2.5) the same as CG. Though the resetting rules are the same, the update rules of the search direction can be viewed as different

linear combinations of the history search direction and the current gradient. The calculation of $\beta_k^{(CG)}$ is based on $\nabla f(\mathbf{x}_k)$ and $\nabla f(\mathbf{x}_{k-1})$, while β_k and γ_k in SDC depend on the restarting condition. Moreover, as mentioned before, the update rule of $\mathbf{u}_k$ with SDC involves an extra term $\gamma_k \frac{\|\mathbf{u}_k\|}{\|\nabla f(\mathbf{x}_k)\|}\nabla f(\mathbf{x}_k)$, which leads to a different combination rule. On the other hand, the performance of nonlinear CG methods strongly depends on step sizes obtained by line search. In general, we cannot use a fixed step size for CG. Hence, we cannot model the trajectory of CG iterations in continuous time as an ODE.

Here we want to emphasize the relationship between the choice of step sizes and the restarting condition. If we want to choose a step size by line search, a basic assumption is that the searching direction $\mathbf{u}_{k+1}$ is a descent direction, or equivalently

$$\langle \mathbf{u}_{k+1}, \nabla f(\mathbf{x}_k)\rangle < 0.$$

The restarting condition essentially makes this assumption to hold. If we do not use restarts, then we may not be able to find a step size by line search when $\langle \mathbf{u}_{k+1}, \nabla f(\mathbf{x}_k)\rangle \geq 0$.

3. SDC from an ODE perspective. In this section, we consider the unconstrained smooth convex optimization problem (1.1). Namely, we consider the case where $h = 0$, $f \in \mathcal{F}_L$. We further assume that f has a unique minimizer $\mathbf{x}^*$ with $\|\mathbf{x}^*\| < \infty$. Moreover, we assume that no restarting condition is applied in Algorithm 2.1 and the step size s_k is fixed to be s.

3.1. SDC in continuous time. By rescaling $\mathbf{v}_k = \sqrt{s}\mathbf{u}_k$, we can write the update rule of $\mathbf{u}_{k+1}$ and $\mathbf{x}_{k+1}$ given by (2.2) and (2.9) as follows:

$$\begin{cases} \dfrac{\mathbf{v}_{k+1} - \mathbf{v}_k}{\sqrt{s}} = -\dfrac{\beta_k}{\sqrt{s}}\mathbf{v}_k - \dfrac{\gamma_k}{\sqrt{s}}\dfrac{\|\mathbf{v}_k\|}{\|\nabla f(\mathbf{x}_k)\|}\nabla f(\mathbf{x}_k) - \nabla f(\mathbf{x}_k), \\ \dfrac{\mathbf{x}_{k+1} - \mathbf{x}_k}{\sqrt{s}} = \mathbf{v}_{k+1}. \end{cases} \tag{3.1}$$

Taking the limit $s \to 0$ in (3.1) and neglecting higher-order terms, we directly have

$$\begin{cases} \dot{\mathbf{v}} = -\nabla f(\mathbf{x}) - \beta(t)\mathbf{v} + \gamma(t)\dfrac{||\mathbf{v}||}{||\nabla f(\mathbf{x})||}\nabla f(\mathbf{x}), \\ \dot{\mathbf{x}} = \mathbf{v}, \end{cases} \tag{3.2}$$

where $\beta(t), \gamma(t) : \mathbb{R}^+ \to \mathbb{R}^+$ can be viewed as rescaled β_k, γ_k in continuous time. Specifically, for FIRE, $\beta(t)$ and $\gamma(t)$ have the following expressions:

$$\beta(t) = \gamma(t) = c_1 e^{-c_2 t}, \tag{3.3}$$

where $c_1, c_2 > 0$ are constants.

Remark 3.1. The continuous time $\beta(t)$ can be derived as follows:

$$\beta(t) = \lim_{s\to 0} \frac{\beta_{\lfloor t/\sqrt{s}\rfloor}}{\sqrt{s}}.$$

We also note that β_k may depend on s. For instance, with $\beta_k = \frac{2\sqrt{\mu s}}{1+\sqrt{\mu s}}$, we have $\beta(t) = 2\sqrt{\mu}$ for a parameter $\mu > 0$.

We can rewrite (3.2) into a second-order ODE:

$$\text{(SDC-ODE)} \qquad \ddot{\mathbf{x}} + \nabla f(\mathbf{x}) + \beta(t)\dot{\mathbf{x}} + \gamma(t)\frac{||\dot{\mathbf{x}}||}{||\nabla f(\mathbf{x})||}\nabla f(\mathbf{x}) = 0.$$

Using the symplectic Euler scheme in molecular dynamics, we can discretize (3.2) by

$$\text{(3.4)} \qquad \begin{cases} \mathbf{v}_{k+1} = \mathbf{v}_k - \sqrt{s}\nabla f(\mathbf{x}_k) - \sqrt{s}\beta(k\sqrt{s})\mathbf{v}_k + \sqrt{s}\gamma(k\sqrt{s})\dfrac{||\mathbf{v}_k||}{||\nabla f(\mathbf{x}_k)||}\nabla f(\mathbf{x}_k), \\ \mathbf{x}_{k+1} = \mathbf{x}_k + \sqrt{s}\mathbf{v}_{k+1}, \end{cases}$$

where $\sqrt{s}$ is the step size. By rescaling $\beta_k = \sqrt{s}\beta(k\sqrt{s})$ and $\gamma_k = \sqrt{s}\gamma(k\sqrt{s})$, (3.4) is equivalent to the update rule (3.1). In other words, we use (SDC-ODE) to model these first-order methods with SDC.

3.2. FISC-ODE with a $\mathcal{O}(1/t^2)$ convergence rate. The Lyapunov function (energy functional) is a powerful tool to analyze the convergence rate of ODE, as mentioned in [27, 28, 25]. But with $\beta(t), \gamma(t)$ specified by (3.3), (SDC-ODE) is hard to directly analyze using Lyapunov's methods. With $t \to \infty$, $\beta(t)$ and $\gamma(t)$ will decay exponentially to zero, but the system will reduce to the following Hamiltonian system:

$$\text{(3.5)} \qquad \ddot{\mathbf{x}} + \nabla f(\mathbf{x}) = 0.$$

We note that the Hamiltonian system (3.5) makes the Hamiltonian $\frac{1}{2}\|\dot{\mathbf{x}}\|^2 + f(\mathbf{x})$ invariant with respect to the time t. As $t \to \infty$, $\mathbf{x}$ following (3.5) may not converge to the minimizer $\mathbf{x}^*$. Hence, it is hard to design a Lyapunov function to analyze the convergence property of the ODE with $\beta(t) = \gamma(t) = \exp(-ct)$.

We hope to choose proper β_k and γ_k to ensure that (SDC-ODE) have certain good properties in Lyapunov analysis. Consider the following Lyapunov function for (SDC-ODE):

$$\text{(3.6)} \qquad \mathcal{E}(t) = \frac{\mu(t)}{2}\|\dot{\mathbf{x}}\|^2 + \frac{1}{2}\|\mathbf{x} - \mathbf{x}^* + \phi(t)\dot{\mathbf{x}}\|^2 + \zeta(t)(f(\mathbf{x}) - f(\mathbf{x}^*)),$$

where $\mu(t), \phi(t)$, and $\zeta(t)$ are mappings $\mathbb{R}^+ \to \mathbb{R}^+$ and $\mathbf{x}^*$ is the unique minimizer of f. The structure of (3.6) is motivated by the Lyapunov function in [27] and [33]. The Lyapunov function in [27] involves terms $\|\mathbf{x}(t) - \mathbf{x}^* + \phi(t)\dot{\mathbf{x}}(t)\|^2$ and $(f(\mathbf{x}(t)) - f(\mathbf{x}^*))$, and [33] introduces an additional term $\|\dot{\mathbf{x}}(t)\|^2$.

We consider a specific selection of $\beta(t), \gamma(t), \mu(t), \phi(t)$, and $\zeta(t)$:

$$\text{(3.7)} \qquad \beta(t) = \frac{r-3}{t}, \quad \gamma(t) = \frac{r}{t}, \quad \mu(t) = \frac{(r-3)t^2}{2(r-1)^2}, \quad \phi(t) = \frac{t}{r-1}, \quad \zeta(t) = \frac{t^2}{2(r-1)},$$

where $r \geq 3$ is a parameter. This renders our proposed (FISC-ODE):

$$\text{(FISC-ODE)} \qquad \ddot{\mathbf{x}} + \frac{r}{t}\dot{\mathbf{x}} + \nabla f(\mathbf{x}) + \frac{r-3}{t}\frac{\|\dot{\mathbf{x}}\|}{\|\nabla f(\mathbf{x})\|}\nabla f(\mathbf{x}) = 0.$$

For the Lyapunov function of (FISC-ODE), we have the following lemma.

LEMMA 1. *Suppose that $f \in \mathcal{F}$ and f has a unique minimizer $\mathbf{x}^*$ with $\|\mathbf{x}^*\| < \infty$ With $\mu(t), \phi(t)$, and $\zeta(t)$ specified in* (3.7)*, the Lyapunov function $\mathcal{E}(t)$ satisfies $\dot{\mathcal{E}}(t) \leq 0$.*

Proof. For simplicity, let $\omega = 1/(r-1)$. Then $(r-3)/(r-1)^2 = \omega - 2\omega^2$, $r = (\omega+1)/\omega$. We can rewrite (FISC-ODE) as

$$\omega t\ddot{\mathbf{x}} = -(1+\omega)\dot{\mathbf{x}} - \omega t\nabla f(\mathbf{x}) - (1-2\omega)\frac{\|\dot{\mathbf{x}}\|}{\|\nabla f(\mathbf{x})\|}\nabla f(\mathbf{x}). \tag{3.8}$$

The convexity of f yields

$$\langle \mathbf{x}-\mathbf{x}^*, \nabla f(\mathbf{x})\rangle \geq \langle \mathbf{x}-\mathbf{x}^*, \nabla f(\mathbf{x})\rangle - f(\mathbf{x}) + f(\mathbf{x}^*) \geq 0. \tag{3.9}$$

The Lyapunov function (3.6) with $\mu(t), \phi(t)$, and $\zeta(t)$ specified in (3.7) writes as

$$\mathcal{E}(t) = \frac{(\omega-2\omega^2)t^2}{4}\|\dot{\mathbf{x}}\|^2 + \frac{1}{2}\left\|\mathbf{x}-\mathbf{x}^* + \omega t\dot{\mathbf{x}}\right\|^2 + \frac{\omega t^2}{2}(f(\mathbf{x}) - f(\mathbf{x}^*)). \tag{3.10}$$

Hence, we obtain

$$\begin{aligned}
2\dot{\mathcal{E}}(t) =& (1-2\omega)t\left\langle \dot{\mathbf{x}}, \omega t\ddot{\mathbf{x}}\right\rangle + (\omega-2\omega^2)t\|\dot{\mathbf{x}}\|^2 + 2\left\langle \mathbf{x}-\mathbf{x}^* + \omega t\dot{\mathbf{x}}, \dot{\mathbf{x}} + \omega\dot{\mathbf{x}} + \omega t\ddot{\mathbf{x}}\right\rangle \\
&+ \omega t^2\left\langle \dot{\mathbf{x}}, \nabla f(\mathbf{x})\right\rangle + 2\omega t(f(\mathbf{x}) - f(\mathbf{x}^*)) \\
=& -(1-2\omega)t\left((1+\omega)\|\dot{\mathbf{x}}\|^2 + \omega t\left\langle \dot{\mathbf{x}}, \nabla f(\mathbf{x})\right\rangle + (1-2\omega)\frac{\|\dot{\mathbf{x}}\|}{\|\nabla f(\mathbf{x})\|}\left\langle \dot{\mathbf{x}}, \nabla f(\mathbf{x})\right\rangle\right) \\
&+ (\omega-2\omega^2)t\|\dot{\mathbf{x}}\|^2 - 2(1-2\omega)\frac{\|\dot{\mathbf{x}}\|}{\|\nabla f(\mathbf{x})\|}\left\langle \mathbf{x}-\mathbf{x}^* + \omega t\dot{\mathbf{x}}, \nabla f(\mathbf{x})\right\rangle \\
&- 2\omega t\left\langle \mathbf{x}-\mathbf{x}^* + \omega t\dot{\mathbf{x}}, \nabla f(\mathbf{x})\right\rangle + \omega t^2\left\langle \dot{\mathbf{x}}, \nabla f(\mathbf{x})\right\rangle + 2\omega t(f(\mathbf{x}) - f(\mathbf{x}^*)) \\
=& -(1-2\omega)t\left(\|\dot{\mathbf{x}}\|^2 + \frac{\|\dot{\mathbf{x}}\|}{\|\nabla f(\mathbf{x})\|}\left\langle \dot{\mathbf{x}}, \nabla f(\mathbf{x})\right\rangle\right) \\
&- 2(1-2\omega)\frac{\|\dot{\mathbf{x}}\|}{\|\nabla f(\mathbf{x})\|}\left\langle \mathbf{x}-\mathbf{x}^*, \nabla f(\mathbf{x})\right\rangle \\
&- 2\omega t\left(\left\langle \mathbf{x}-\mathbf{x}^*, \nabla f(\mathbf{x})\right\rangle - f(\mathbf{x}) + f(\mathbf{x}^*)\right) \leq 0,
\end{aligned}$$

where the second equality is due to (3.8) and the last inequality takes (3.9). ∎

Based on Lemma 1, we have the following convergence rate of (FISC-ODE).

THEOREM 3.2 (the $\mathcal{O}(t^{-2})$ convergence rate of FISC-ODE). *Suppose that $f \in \mathcal{F}$ and f has a unique minimizer $\mathbf{x}^*$ with $\|\mathbf{x}^*\| < \infty$. For any $r \geq 3$, let $\mathbf{x}(t)$ be the solution to (FISC-ODE) with initial conditions $\mathbf{x}(0) = \mathbf{x}_0$ and $\dot{\mathbf{x}}(0) = 0$. Then, for $t > 0$, we have*

$$f(\mathbf{x}(t)) - f(\mathbf{x}^*) \leq \frac{(r-1)\|\mathbf{x}_0 - \mathbf{x}^*\|^2}{t^2}.$$

Proof. The definition (3.6) gives

$$\mathcal{E}(t) \geq \frac{t^2(f(\mathbf{x}(t)) - f(\mathbf{x}^*))}{2(r-1)}.$$

Since $\mathcal{E}(t)$ is nonincreasing from Lemma 1, we obtain

$$f(\mathbf{x}(t)) - f(\mathbf{x}^*) \leq \frac{2(r-1)\mathcal{E}(t)}{t^2} \leq \frac{2(r-1)\mathcal{E}(0)}{t^2} = \frac{(r-1)\|\mathbf{x}_0 - \mathbf{x}^*\|^2}{t^2} = O(t^{-2}),$$

which completes the proof. ∎

Now, rewriting (FISC-ODE) into a first-order ODE system and discretizing it with the symplectic Euler scheme, we can directly recover the update rule of FISC (2.12) with $l_k = k$. We can also discretize (FISC-ODE) with techniques analogous to Nesterov's accelerated method, and then the update rule of FISC-nes (2.11) is recovered.

3.3. Comparison with other first-order methods with ODE interpretations. If we take $r = 3$, then (FISC-ODE) turns out to be

$$\ddot{\mathbf{x}} + \frac{3}{t}\dot{\mathbf{x}} + \nabla f(\mathbf{x}) = 0. \tag{Nesterov-ODE}$$

[25] used this ODE for modeling Nesterov's accelerated method.

Dropping the term $(r-3)\|\dot{\mathbf{x}}\|\nabla f(\mathbf{x})/(t\|\nabla f(\mathbf{x})\|)$, (FISC-ODE) becomes

$$\ddot{\mathbf{x}} + \frac{r}{t}\dot{\mathbf{x}} + \nabla f(\mathbf{x}) = 0, \tag{HF-nes-ODE}$$

which is the high friction version of (Nesterov-ODE) in [25] with $r \geq 3$.

Under the special case $r = 3$, the coefficient of the term $\frac{||\mathbf{u}_k||}{||\nabla f(\mathbf{x}_k)||}\nabla f(\mathbf{x}_k)$ in (2.11) turns out to be 0. If no restarting condition is applied and the step size is fixed, FISC-nes becomes the Nesterov's accelerated method. With restarts and a fixed step size, FISC-nes recovers Nesterov's accelerated method with gradient restarting [22]. Therefore, we can view FISC-nes as an extension of the restarting Nesterov's accelerated method. Furthermore, numerical experiments indicate that a proper choice of r leads to extra acceleration in Nesterov's accelerated method.

We also observe that the ODE model of the heavy-ball method is given by

$$\ddot{\mathbf{x}} + \beta\dot{\mathbf{x}} + \nabla f(\mathbf{x}) = 0, \tag{HB-ODE}$$

where β is a constant. The convergence rate of (HB-ODE) is an open problem for a general convex f. Indeed, it is shown in subsection 4.6 in [15] that the heavy-ball method does not converge for general convex f.

In summary, (Nesterov-ODE), (HF-nes-ODE), and (HB-ODE) can be viewed as specific examples of (SDC-ODE) with different choices of $\beta(t)$ and $\gamma(t)$.

4. Convergence analysis. In this section, we analyze the global convergence of methods with SDC for general unconstrained smooth optimization problems and the convergence of FISC-PM for composite optimization problems. In both cases, we assume that the target function f is bounded from below.

4.1. The global convergence of methods with SDC. In this subsection, we show the global convergence of methods with SDC and explain why we use (2.1) as our restarting condition. We consider the case where the objective function is smooth, i.e., $h = 0$ in (1.1). Namely, f is L-smooth. We begin with the following lemma.

LEMMA 2. *Suppose that f is differentiable, $\mathbf{u}_{k+1}$ is updated by* (2.2) *or* (2.5) *depending on the restarting condition using φ_k, and β_k and γ_k satisfy* (2.3)*. Then, for any integer $k \geq 0$, we have*

$$\langle \mathbf{u}_{k+1}, -\nabla f(\mathbf{x}_k)\rangle \geq \|\nabla f(\mathbf{x}_k)\|^2. \tag{4.1}$$

Proof. If $\varphi_k \geq 0$, then we update $\mathbf{u}_{k+1}$ by (2.2). Hence,

$$\begin{aligned}&\langle \mathbf{u}_{k+1}, -\nabla f(\mathbf{x}_k)\rangle \\ &\quad = (1-\beta_k)\langle \mathbf{u}_k, -\nabla f(\mathbf{x}_k)\rangle + \gamma_k\|\mathbf{u}_k\|\|\nabla f(\mathbf{x}_k)\| + \|\nabla f(\mathbf{x}_k)\|^2 \geq \|\nabla f(\mathbf{x}_k)\|^2.\end{aligned}$$

If $\varphi_k < 0$, we reset $\mathbf{u}_{k+1} = -\nabla f(\mathbf{x}_k)$ and $\langle \mathbf{u}_{k+1}, -\nabla f(\mathbf{x}_k)\rangle = \|\nabla f(\mathbf{x}_k)\|^2$. ∎

According to Lemma 2, we have the following lower bound:

$$\frac{\langle \mathbf{u}_{k+1}, -\nabla f(\mathbf{x}_k)\rangle}{\|\mathbf{u}_{k+1}\|\|\nabla f(\mathbf{x}_k)\|} \geq \frac{\|\nabla f(\mathbf{x}_k)\|}{\|\mathbf{u}_{k+1}\|}. \tag{4.2}$$

From our assumption that f is bounded from below, there exists s_k satisfying the Armijo conditions (2.7) or the nonmonotone Armijo conditions (2.8), according to Lemma 1.1 in [32].

To ensure the global convergence rate, we require that for sufficiently large k,

$$\|\mathbf{u}_{k+1}\| \leq C\|\nabla f(\mathbf{x}_k)\|$$

for some constant C. This condition can be hard to meet for a general choice of β_k and γ_k. Hence, we add two restarting conditions:

$$d_f\|\nabla f(\mathbf{x}_k)\| \geq \|\nabla f(\mathbf{x}_{k-1})\|, \tag{4.3}$$

$$n_k \leq K, \tag{4.4}$$

where $K \in \mathbb{N}$, $d_f > 1$, and n_k is the number of iterations since the last restart. Namely, we restart our system if at least one of the conditions (2.1), (4.3), and (4.4) is not satisfied. If we set d_f and K large enough in practice, conditions (4.3) and (4.4) will seldom be violated. Equipped with restarting conditions (4.3) and (4.4), the system will restart at least once in K consecutive iterations and $\|\nabla f(\mathbf{x}_k)\|$ will not drop too rapidly. We then introduce the following lemma.

LEMMA 3. *Suppose that the conditions of Lemma* 2 *are satisfied. $K' \leq K$ is an integer and both* (2.1) *and* (4.3) *hold for $1 \leq k \leq K'$. Then, $\frac{\|\mathbf{u}_{k+1}\|}{\|\nabla f(\mathbf{x}_k)\|}$ is upper bounded for all $0 \leq k \leq K'$ regardless of the initial value $\mathbf{x}_0$.*

Proof. Let $\lambda_k = \frac{\|\mathbf{u}_{k+1}\|}{\|\nabla f(\mathbf{x}_k)\|}, \xi_k = \frac{\|\mathbf{u}_k\|}{\|\nabla f(\mathbf{x}_k)\|}$. Specifically, $\lambda_0 = 1$. Based on (4.3), we have $\xi_k \leq d_f\lambda_{k-1}$. Hence,

$$\begin{aligned}
\lambda_k^2 = \frac{1}{\|\nabla f(\mathbf{x}_k)\|^2}\Bigg[&(1-\beta_k)^2\|\mathbf{u}_k\|^2 + \left(1+\gamma_k\frac{\|\mathbf{u}_k\|}{\|\nabla f(\mathbf{x}_k)\|}\right)^2\|\nabla f(\mathbf{x}_k)\|^2 \\
&+ 2(1-\beta_k)\left(1+\gamma_k\frac{\|\mathbf{u}_k\|}{\|\nabla f(\mathbf{x}_k)\|}\right)\langle \mathbf{u}_k, -\nabla f(\mathbf{x}_k)\rangle\Bigg] \\
&\leq (1-\beta_k)^2\xi_k^2 + (1+\gamma_k\xi_k)^2 + 2(1-\beta_k)(1+\gamma_k\xi_k)\xi_k \\
&\leq (1-\beta_k)^2 d_f^2\lambda_{k-1}^2 + (1+\gamma_k d_f\lambda_{k-1})^2 + 2(1-\beta_k)(1+\gamma_k d_f\lambda_{k-1})d_f\lambda_{k-1} \\
&\leq d_f^2\lambda_{k-1}^2 + 2(1+d_f\lambda_{k-1})^2 = 4d_f^2\lambda_{k-1}^2 + 4d_f\lambda_{k-1} + 2.
\end{aligned}$$

Consider a sequence $\{\tilde{\lambda}_k\}_{k=0}$ satisfying $\tilde{\lambda}_k^2 = 4d_f^2\tilde{\lambda}_{k-1}^2 + 4d_f\tilde{\lambda}_{k-1} + 2$ and $\tilde{\lambda}_0 = 1$. Because $d_f > 1$, it is obvious that $\tilde{\lambda}_k$ is increasing with respect to k. Then,

$$\lambda_k \leq \tilde{\lambda}_k \leq \tilde{\lambda}_{K'} \leq \tilde{\lambda}_K, \quad 0 \leq k \leq K',$$

which completes the proof. ∎

Lemmas 2 and 3 guarantee that the direction assumption in [32] holds. Namely, there exist positive constants c_1 and c_2 such that

$$\langle \mathbf{u}_{k+1}, \nabla f(\mathbf{x}_k)\rangle \leq -c_1\|\nabla f(\mathbf{x}_k)\|^2, \quad \|\mathbf{u}_{k+1}\| \leq c_2\|\nabla f(\mathbf{x}_k)\|. \tag{4.5}$$

Consider the sequence $\{\mathbf{x}_k\}$ given by Algorithm 2.1 with extra restarting conditions (4.3) and (4.4). We further assume that the step size s_k is attained by the nonmonotone line search. Note that $f(\mathbf{x})$ is bounded from below, the direction assumption (4.5) holds, and the step sizes satisfy the nonmonotone Armijo conditions. According to Theorem 2.2 in [32], we obtain

$$\lim_{k\to\infty} \inf \|\nabla f(\mathbf{x}_k)\| = 0.$$

Moreover, if $\eta_{max} < 1$ (η_{max} is a parameter for the nonmonotone line search), then we have

$$\lim_{k\to\infty} \|\nabla f(\mathbf{x}_k)\| = 0,$$

which indicates the global convergence of first-order methods with SDC.

4.2. The $\mathcal{O}(1/k^2)$ convergence rate of FISC-PM. We analyze the convergence of FISC-PM for the composite optimization problem (1.1) with a unique minimizer $\mathbf{x}^*$. It is assumed that $\psi \in \mathcal{F}_L$ is bounded from below. We consider the case that the step size is fixed to be $0 < s \le 1/L$ and no restarts are used, i.e., the sequences $\{\mathbf{x}_k\}$ and $\{\mathbf{y}_k\}$ are merely updated by (2.16) and (2.17). In other words, β_k, γ_k are specified by (2.4) with $l_k = k$. We introduce the following discrete-time Lyapunov function $\mathcal{E}(k)$:

$$\begin{aligned}\mathcal{E}(k) = 2\left\|\mathbf{x}_k - \mathbf{x}^* + \frac{k-1}{r-1}(\mathbf{x}_k - \mathbf{x}_{k-1})\right\|^2 + \frac{2(k+r-2)^2 s}{r-1}(f(\mathbf{x}_k) - f(\mathbf{x}^*)) \\ + \frac{(r-3)(k-1)^2}{(r-1)^2}\|\mathbf{x}_k - \mathbf{x}_{k-1}\|^2.\end{aligned} \tag{4.6}$$

The discrete-time function $\mathcal{E}(k)$ corresponds to the continuous-time Lyapunov function (3.10). We introduce a basic inequality in convex optimization.

LEMMA 4. *Consider a convex function of the form $f(\mathbf{x}) = \psi(\mathbf{x}) + h(\mathbf{x})$, where $\psi \in \mathcal{F}_L$ and h is convex. For any $0 < s \le 1/L$ and $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, we have*

$$f(\mathbf{y} - sG_s(\mathbf{y})) \le f(\mathbf{x}) + G_s(\mathbf{y})^T(\mathbf{y} - \mathbf{x}) - \frac{s}{2}\|G_s(\mathbf{y})\|^2. \tag{4.7}$$

Proof. Define $\mathbf{v} = G_s(\mathbf{y}) - \nabla\psi(\mathbf{y})$. Then, we have

$$\begin{aligned} f(\mathbf{y} - sG_s(\mathbf{y})) &\le \psi(\mathbf{y}) - s\nabla\psi(\mathbf{y})^T G_s(\mathbf{y}) + \frac{Ls^2}{2}\|G_s(\mathbf{y})\|_2^2 + h(\mathbf{y} - sG_s(\mathbf{y})) \\ &\le \psi(\mathbf{y}) - s\nabla\psi(\mathbf{y})^T G_s(\mathbf{y}) + \frac{s}{2}\|G_s(\mathbf{y})\|_2^2 + h(\mathbf{y} - sG_s(\mathbf{y})) \\ &\le \psi(\mathbf{x}) + \nabla\psi(\mathbf{y})^T(\mathbf{y} - \mathbf{x}) - s\nabla\psi(\mathbf{y})^T G_s(\mathbf{y}) + \frac{s}{2}\|G_s(\mathbf{y})\|_2^2 \\ &\quad + h(\mathbf{x}) + v^T(\mathbf{y} - \mathbf{x} - sG_s(\mathbf{y})) \\ &= \psi(\mathbf{x}) + h(\mathbf{x}) + G_s(\mathbf{y})^T(\mathbf{y} - \mathbf{x}) - \frac{s}{2}\|G_s(\mathbf{y})\|_2^2, \end{aligned} \tag{4.8}$$

where the first inequality utilizes that $\psi \in \mathcal{F}_L$, the second inequality applies $s \le 1/L$, and the third inequality follows from the convexity of g and h and $\mathbf{v} \in \partial h(\mathbf{y} - sG_s(\mathbf{y}))$. ∎

Based on the basic inequality (4.7), we give the following estimation.

LEMMA 5. *Suppose that $f(\mathbf{x}) = \psi(\mathbf{x}) + h(\mathbf{x})$, where $\psi \in \mathcal{F}_L$ and h is convex. Assume that f has a unique minimizer $\mathbf{x}^*$ with $\|\mathbf{x}^*\| < \infty$. The discrete Lyapunov function $\mathcal{E}(k)$ given by* (4.6) *satisfies*

$$\mathcal{E}(k)-\mathcal{E}(k-1) \le \alpha(\phi_{k-1}-2)\|\Delta\mathbf{x}_{k-1}\|^2-\alpha\phi_k\|\Delta\mathbf{x}_k\|^2-\frac{2s}{r-1}(f(\mathbf{x}_{k-1})-f(\mathbf{x}^*)), \tag{4.9}$$

where

$$\alpha = \frac{r-3}{r-1},\ \phi_k = 2k+r-3,\ \Delta\mathbf{x}_k = \mathbf{x}_k - \mathbf{x}_{k-1}. \tag{4.10}$$

Proof. For simplicity, we denote

$$\mathbf{r}_k = \frac{\|\Delta\mathbf{x}_k\|}{\|G_s(\mathbf{x}_k)\|}G_s(\mathbf{x}_k),\ \xi_k = \frac{k+r-2}{r-1},\ \nu_k = \frac{2(k+r-2)(k+r-4)}{r-1}$$

and introduce two auxiliary variables $\mathbf{z}_k$ and $\mathbf{w}_k$ defined by

$$\mathbf{z}_k = \mathbf{x}_k + \frac{k-1}{r-1}\Delta\mathbf{x}_k,\ \mathbf{w}_k = \mathbf{z}_k + \mathbf{z}_{k-1} - \mathbf{x}_k - \mathbf{x}_{k-1}. \tag{4.11}$$

We can also write $\mathbf{z}_{k-1}$ in the following way:

$$\begin{aligned}\mathbf{z}_{k-1} &= \mathbf{x}_{k-1} + \frac{k-2}{r-1}\Delta\mathbf{x}_{k-1} = \frac{k+r-2}{r-1}\left(\mathbf{x}_{k-1} + \frac{k-2}{k+r-2}\Delta\mathbf{x}_{k-1}\right) - \frac{k-1}{r-1}\mathbf{x}_{k-1}\\ &= \frac{k+r-2}{r-1}\left(\mathbf{y}_{k-1} + \frac{r-3}{k+r-2}\mathbf{r}_{k-1}\right) - \frac{k-1}{r-1}\mathbf{x}_{k-1} = \xi_k\mathbf{y}_{k-1} + \alpha\mathbf{r}_{k-1} - \frac{k-1}{r-1}\mathbf{x}_{k-1}.\end{aligned} \tag{4.12}$$

The update rule (2.16) and (2.17) can be written as

$$\frac{k-2}{r-1}\Delta\mathbf{x}_{k-1} - \alpha\mathbf{r}_{k-1} = \xi_k\left(\Delta\mathbf{x}_k + sG_s(\mathbf{y}_{k-1})\right). \tag{4.13}$$

Based on (4.12) and (4.13), we can write

$$\begin{aligned}\mathbf{z}_k - \mathbf{z}_{k-1} &= \Delta\mathbf{x}_k + \frac{k-1}{r-1}\Delta\mathbf{x}_k - \frac{k-2}{r-1}\Delta\mathbf{x}_{k-1}\\ &= \xi_k\Delta\mathbf{x}_k - \frac{k-2}{r-1}\Delta\mathbf{x}_{k-1} = -\alpha\mathbf{r}_{k-1} - \xi_k sG_s(\mathbf{y}_{k-1}),\\ \mathbf{z}_k + \mathbf{z}_{k-1} &= \mathbf{z}_k - \mathbf{z}_{k-1} + 2\mathbf{z}_{k-1}\\ &= -\alpha\mathbf{r}_{k-1} - \xi_k sG_s(\mathbf{y}_{k-1}) + 2\xi_k\mathbf{y}_{k-1} - \frac{2(k-1)}{r-1}\mathbf{x}_{k-1} + 2\alpha\mathbf{r}_{k-1}\\ &= -\xi_k sG_s(\mathbf{y}_{k-1}) + 2\xi_k\mathbf{y}_{k-1} - \frac{2(k-1)}{r-1}\mathbf{x}_{k-1} + \alpha\mathbf{r}_{k-1}.\end{aligned} \tag{4.14}$$

Using (4.11), (4.13), and the fact $\frac{k-1}{r-1} + \xi_k = \frac{2k+r-3}{r-1} = \frac{\phi_k}{r-1}$ yields

$$\begin{aligned}\mathbf{w}_k &= \frac{k-1}{r-1}\Delta\mathbf{x}_k + \frac{k-2}{r-1}\Delta\mathbf{x}_{k-1} = \frac{k-1}{r-1}\Delta\mathbf{x}_k + \alpha\mathbf{r}_{k-1} + \xi_k\left(\Delta\mathbf{x}_k + sG_s(\mathbf{y}_{k-1})\right)\\ &= \frac{\phi_k}{r-1}\Delta\mathbf{x}_k + \alpha\mathbf{r}_{k-1} + \xi_k sG_s(\mathbf{y}_{k-1}).\end{aligned} \tag{4.15}$$

We now analyze the difference between $2\|\mathbf{x}_k - \mathbf{x}^* + \frac{k-1}{r-1}\Delta\mathbf{x}_k\|^2$ in $\mathcal{E}(k)$:

(4.16)
$$\begin{aligned}
&2\left\|\mathbf{x}_k - \mathbf{x}^* + \frac{k-1}{r-1}\Delta\mathbf{x}_k\right\|^2 - 2\left\|\mathbf{x}_{k-1} - \mathbf{x}^* + \frac{k-2}{r-1}\Delta\mathbf{x}_{k-1}\right\|^2 \\
&= 2\|\mathbf{z}_k - \mathbf{x}^*\|^2 - 2\|\mathbf{z}_{k-1} - \mathbf{x}^*\|^2 = 2(\mathbf{z}_k - \mathbf{z}_{k-1})^T(\mathbf{z}_k + \mathbf{z}_{k-1} - 2\mathbf{x}^*) \\
&= -2\xi_k s G_s(\mathbf{y}_{k-1})^T(\mathbf{z}_k + \mathbf{z}_{k-1} - 2\mathbf{x}^*) - 2\alpha\mathbf{r}_{k-1}^T(\mathbf{z}_k + \mathbf{z}_{k-1} - 2\mathbf{x}^*) \\
&= -2\xi_k s G_s(\mathbf{y}_{k-1})^T(\mathbf{z}_k + \mathbf{z}_{k-1} - 2\mathbf{x}^*) - 2\alpha\mathbf{r}_{k-1}^T\left(\mathbf{x}_k + \mathbf{x}_{k-1} - 2\mathbf{x}^*\right) - 2\alpha\mathbf{r}_{k-1}^T\mathbf{w}_k.
\end{aligned}$$

Then, the difference between $\frac{(r-3)(k-1)^2}{(r-1)^2}\|\Delta\mathbf{x}_k\|^2$ in $\mathcal{E}(k)$ is calculated by

(4.17)
$$\begin{aligned}
&\frac{(r-3)(k-1)^2}{(r-1)^2}\|\Delta\mathbf{x}_k\|^2 - \frac{(r-3)(k-2)^2}{(r-1)^2}\|\Delta\mathbf{x}_{k-1}\|^2 \\
&= (r-3)(\|\mathbf{z}_k - \mathbf{x}_k\|^2 - \|\mathbf{z}_{k-1} - \mathbf{x}_{k-1}\|^2) = (r-3)(\mathbf{z}_k - \mathbf{z}_{k-1} - \Delta\mathbf{x}_k)^T\mathbf{w}_k \\
&= (r-3)\left(-\alpha\mathbf{r}_{k-1} - \xi_k s G_s(\mathbf{y}_{k-1}) - \Delta\mathbf{x}_k\right)^T\mathbf{w}_k.
\end{aligned}$$

By using (4.16) and (4.17), we can split $\mathcal{E}(k) - \mathcal{E}(k-1)$ into three parts:

(4.18)
$$\begin{aligned}
&\mathcal{E}(k) - \mathcal{E}(k-1) \\
&= -2\xi_k G_s(\mathbf{y}_{k-1})^T(\mathbf{z}_k + \mathbf{z}_{k-1} - 2\mathbf{x}^*) - 2\alpha\mathbf{r}_{k-1}^T\left(\mathbf{x}_k + \mathbf{x}_{k-1} - 2\mathbf{x}^*\right) \\
&\quad - 2\alpha\mathbf{r}_{k-1}^T\mathbf{w}_k - (r-3)\left(\alpha\mathbf{r}_{k-1}^T\mathbf{w}_k + \xi_k s G_s(\mathbf{y}_{k-1})^T\mathbf{w}_k + \Delta\mathbf{x}_k^T\mathbf{w}_k\right) \\
&\quad + \frac{2(k+r-2)^2 s}{r-1}(f(\mathbf{x}_k) - f(\mathbf{x}^*)) - \frac{2(k+r-3)^2 s}{r-1}(f(\mathbf{x}_{k-1}) - f(\mathbf{x}^*)) \\
&= -(r-3)(\mathbf{r}_{k-1} + \Delta\mathbf{x}_k)^T\mathbf{w}_k - 2\alpha\mathbf{r}_{k-1}^T\left(\mathbf{x}_k + \mathbf{x}_{k-1} - 2\mathbf{x}^*\right) \\
&\quad - 2\xi_k s G_s(\mathbf{y}_{k-1})^T(\mathbf{z}_k + \mathbf{z}_{k-1} - 2\mathbf{x}^*) - (r-3)\xi_k s G_s(\mathbf{y}_{k-1})^T\mathbf{w}_k \\
&\quad + \frac{2(k+r-2)^2 s}{r-1}(f(\mathbf{x}_k) - f(\mathbf{x}^*)) - \frac{2(k+r-3)^2 s}{r-1}(f(\mathbf{x}_{k-1}) - f(\mathbf{x}^*)).
\end{aligned}$$

The quantities in the last three rows of (4.18) are denoted as L_1, L_2, and L_3, respectively. It follows from (4.15) and $r - 3 = \alpha(r-1)$ that

(4.19)
$$\begin{aligned}
&L_1 + 4\alpha\mathbf{r}_{k-1}^T(\mathbf{x}_{k-1} - \mathbf{x}^*) = -(r-3)(\mathbf{r}_{k-1} + \Delta\mathbf{x}_k)^T\mathbf{w}_k - 2\alpha\mathbf{r}_{k-1}^T\Delta\mathbf{x}_k \\
&= -(r-3)(\mathbf{r}_{k-1} + \Delta\mathbf{x}_k)^T\left(\frac{\phi_k}{r-1}\Delta\mathbf{x}_k + \alpha\mathbf{r}_{k-1} + \xi_k s G_s(\mathbf{y}_{k-1})\right) - 2\alpha\mathbf{r}_{k-1}^T\Delta\mathbf{x}_k \\
&= -(r-3)\xi_k\mathbf{r}_{k-1}^T s G_s(\mathbf{y}_{k-1}) - (r-3)\xi_k\Delta\mathbf{x}_k^T s G_s(\mathbf{y}_{k-1}) \\
&\quad - \alpha(\phi_k\|\Delta\mathbf{x}_k\|^2 + (r-3)\|\mathbf{r}_{k-1}\|^2)) - 2(r-3)\xi_k\mathbf{r}_{k-1}^T\Delta\mathbf{x}_k \\
&= -\alpha\left(\phi_k\|\Delta\mathbf{x}_k\|^2 + (r-3)\|\mathbf{r}_{k-1}\|^2 + 2(r-1)\xi_k\mathbf{r}_{k-1}^T(\Delta\mathbf{x}_k + s G_s(\mathbf{y}_{k-1}))\right) + \bar{L}_1,
\end{aligned}$$

where

(4.20)
$$\begin{aligned}
\bar{L}_1 &= (r-3)\xi_k\mathbf{r}_{k-1}^T s G_s(\mathbf{y}_{k-1}) - (r-3)\xi_k\Delta\mathbf{x}_k^T s G_s(\mathbf{y}_{k-1}) \\
&= (r-3)\xi_k s G_s(\mathbf{y}_{k-1})^T(\mathbf{r}_{k-1} - \Delta\mathbf{x}_k).
\end{aligned}$$

Utilizing (4.13) and $\|\mathbf{r}_{k-1}\| = \|\Delta\mathbf{x}_{k-1}\|$, we obtain

$$\begin{aligned}
&\phi_k\|\Delta\mathbf{x}_k\|^2 + (r-3)\|\mathbf{r}_{k-1}\|^2 + 2(r-1)\xi_k\mathbf{r}_{k-1}^T(\Delta\mathbf{x}_k + sG_s(\mathbf{y}_{k-1}))\\
&= \phi_k\|\Delta\mathbf{x}_k\|^2 + (r-3)\|\mathbf{r}_{k-1}\|^2 + 2\mathbf{r}_{k-1}^T((k-2)\Delta\mathbf{x}_{k-1} - (r-3)\mathbf{r}_{k-1})\\
&= \phi_k\|\Delta\mathbf{x}_k\|^2 - (r-3)\|\mathbf{r}_{k-1}\|^2 + 2(k-2)\mathbf{r}_{k-1}^T\Delta\mathbf{x}_{k-1}\\
&\geq \phi_k\|\Delta\mathbf{x}_k\|^2 - (2k-r-7)\|\mathbf{r}_{k-1}\|^2 = \phi_k\|\Delta\mathbf{x}_k\|^2 - (\phi_{k-1}-2)\|\Delta\mathbf{x}_{k-1}\|^2.
\end{aligned} \tag{4.21}$$

The last inequality even holds when $k = 1$ because $\mathbf{r}_0 = \Delta\mathbf{x}_0 = 0$. By setting $\mathbf{y} = \mathbf{x}_{k-1}$, $\mathbf{x} = \mathbf{x}^*$ in the basic inequality (4.7), we have

$$\begin{aligned}
&\frac{\|G_s(\mathbf{x}_{k-1})\|}{\|\Delta\mathbf{x}_k\|}\mathbf{r}_{k-1}^T(\mathbf{x}_{k-1} - \mathbf{x}^*) = G_s(\mathbf{x}_{k-1})^T(\mathbf{x}_{k-1} - \mathbf{x}^*)\\
&\geq f(\mathbf{x}_{k-1} - sG_s(\mathbf{x}_{k-1})) - f(\mathbf{x}^*) + \frac{s}{2}\|G_s(\mathbf{x}_{k-1})\|^2 \geq 0.
\end{aligned} \tag{4.22}$$

Substituting inequalities (4.21) and (4.22) in (4.19) yields

$$L_1 \leq -\alpha\phi_k\|\Delta\mathbf{x}_k\|^2 + \alpha(\phi_{k-1}-2)\|\Delta\mathbf{x}_{k-1}\|^2 + \bar{L}_1. \tag{4.23}$$

From the definition of $\mathbf{w}_k$ and (4.14), we obtain

$$\begin{aligned}
&2(\mathbf{z}_k + \mathbf{z}_{k-1} - 2\mathbf{x}^*) + (r-3)\mathbf{w}_k\\
&= 2(\mathbf{z}_k + \mathbf{z}_{k-1} - 2\mathbf{x}^*) + (r-3)((\mathbf{z}_k + \mathbf{z}_{k-1} - 2\mathbf{x}^*) - (\mathbf{x}_k + \mathbf{x}_{k-1} - 2\mathbf{x}^*))\\
&= (r-1)\left(-\xi_k sG_s(\mathbf{y}_{k-1}) + \xi_k 2\mathbf{y}_{k-1} - \frac{2(k-1)}{r-1}\mathbf{x}_{k-1} + \alpha\mathbf{r}_{k-1} - 2\mathbf{x}^*\right)\\
&\quad - 2(r-3)(\mathbf{x}_{k-1} - \mathbf{x}^*) - (r-3)\Delta\mathbf{x}_k\\
&= (r-1)\left(-\xi_k sG_s(\mathbf{y}_{k-1}) + 2\xi_k\mathbf{y}_{k-1} - \frac{2(k-1)}{r-1}\mathbf{x}_{k-1} - 2\mathbf{x}^*\right)\\
&\quad - 2(r-3)(\mathbf{x}_{k-1} - \mathbf{x}^*) - (r-3)(\Delta\mathbf{x} - \mathbf{r}_{k-1}).
\end{aligned} \tag{4.24}$$

The above estimation implies

$$\begin{aligned}
L_2 &= -\xi_k sG_s(\mathbf{y}_{k-1})^T(2(\mathbf{z}_k + \mathbf{z}_{k-1} - 2\mathbf{x}^*) + (r-3)\mathbf{w}_k)\\
&= -(r-1)\xi_k sG_s(\mathbf{y}_{k-1})^T\left(-\xi_k sG_s(\mathbf{y}_{k-1}) + 2\xi_k\mathbf{y}_{k-1} - \frac{2(k-1)}{r-1}\mathbf{x}_{k-1} - 2\mathbf{x}^*\right)\\
&\quad + 2(r-3)\xi_k sG_s(\mathbf{y}_{k-1})^T(\mathbf{x}_{k-1} - \mathbf{x}^*) + (r-3)\xi_k sG_s(\mathbf{y}_{k-1})^T(\Delta\mathbf{x}_k - \mathbf{r}_{k-1}).
\end{aligned} \tag{4.25}$$

Finally, we compute L_3. Note that $\mathbf{x}_k = \mathbf{y}_{k-1} - sG_s(\mathbf{y}_{k-1})$. Taking $\mathbf{y} = \mathbf{y}_{k-1}$, $\mathbf{x} = \mathbf{x}_k$, or $\mathbf{x}^*$ in the basic inequality (4.7) gives

$$\begin{aligned}
f(\mathbf{x}_k) &\leq f(\mathbf{x}_{k-1}) + G_s(\mathbf{y}_{k-1})^T(\mathbf{y}_{k-1} - \mathbf{x}_{k-1}) - \frac{s}{2}\|G_s(\mathbf{y}_{k-1})\|^2,\\
f(\mathbf{x}_k) &\leq f(\mathbf{x}^*) + G_s(\mathbf{y}_{k-1})^T(\mathbf{y}_{k-1} - \mathbf{x}^*) - \frac{s}{2}\|G_s(\mathbf{y}_{k-1})\|^2.
\end{aligned} \tag{4.26}$$

Based on the above inequalities, we observe that

(4.27)
$$\begin{aligned}
&L_3 + \frac{2s}{r-1}(f(\mathbf{x}_{k-1}) - f(\mathbf{x}^*)) \\
&= \frac{2(k+r-2)^2 s}{r-1}(f(\mathbf{x}_k) - f(\mathbf{x}^*)) - \frac{2(k+r-2)(k+r-4)s}{r-1}(f(\mathbf{x}_{k-1}) - f(\mathbf{x}^*)) \\
&= 4\xi_k s(f(\mathbf{x}_k) - f(\mathbf{x}^*)) + \nu_k s(f(\mathbf{x}_k) - f(\mathbf{x}_{k-1})) \\
&\le 4\xi_k \left(sG_s(y_{k-1})^T(\mathbf{y}_{k-1} - \mathbf{x}^*) - \frac{s}{2}\|G_s(\mathbf{y}_{k-1})\|^2 \right) \\
&\quad + \nu_k \left(G_s(\mathbf{y}_{k-1})^T(\mathbf{y}_{k-1} - \mathbf{x}^*) - \frac{s}{2}\|G_s(\mathbf{y}_{k-1})\|^2 \right) = \bar{L}_3.
\end{aligned}$$

Note that $4\xi_k + \nu_k = \frac{2(k+r-2)^2}{r-1} = \frac{(r-1)\xi_k^2}{8}$. $\bar{L}_3$ can be rewritten into

(4.28)
$$\begin{aligned}
\bar{L}_3 &= (4\xi_k + \nu_k)s \left(G_s(\mathbf{y}_{k-1})^T\mathbf{y}_{k-1} - \frac{s}{2}\|G_s(\mathbf{y}_{k-1})\|^2 \right) - sG_s(\mathbf{y}_{k-1})^T(4\xi_k\mathbf{x}^* + \nu_k\mathbf{x}_{k-1}) \\
&= 2(r-1)\xi_k^2 s \left(G_s(\mathbf{y}_{k-1})^T\mathbf{y}_{k-1} - \frac{s}{2}\|G_s(\mathbf{y}_{k-1})\|^2 \right) \\
&\quad - sG_s(\mathbf{y}_{k-1})^T \left(2((r-1)-(r-3))\xi_k\mathbf{x}^* + 2((k-1)-(r-3))\xi_k\mathbf{x}_{k-1} \right) \\
&= (r-1)\xi_k s G_s(\mathbf{y}_{k-1})^T \left(2\xi_k\mathbf{y}_{k-1} - \frac{2(k-1)}{r-1}\mathbf{x}_{k-1} - 2\mathbf{x}^* - \xi_k s G_s(\mathbf{y}_{k-1}) \right) \\
&\quad + 2(r-3)\xi_k s G_s(\mathbf{y}_{k-1})^T(\mathbf{x}^* - \mathbf{x}_{k-1}).
\end{aligned}$$

Together with (4.20) and (4.25), we have

$$\bar{L}_1 + L_2 + \bar{L}_3 = 0. \tag{4.29}$$

Therefore, substituting (4.23), (4.27), and (4.29) in (4.18) renders (4.9). ∎

Based on Lemma 5, we have the following estimation of $\mathcal{E}(k)$.

LEMMA 6 (discrete Lyapunov analysis of FISC-PM). *Suppose that the objective function is $f(\mathbf{x}) = \psi(\mathbf{x}) + h(\mathbf{x})$, where $\psi \in \mathcal{F}_L$ and h is convex. Assume that f has a unique minimizer $\mathbf{x}^*$ with $\|\mathbf{x}^*\| < \infty$. The Lyapunov function $\mathcal{E}(k)$ defined in* (4.6) *satisfies*

$$\mathcal{E}(k) \le \mathcal{E}(0) - \frac{2s}{r-1}(f(\mathbf{x}_0) - f(\mathbf{x}^*)). \tag{4.30}$$

Proof. Note that $\Delta\mathbf{x}_0 = \mathbf{x}_0 - \mathbf{x}_{-1} = 0$. Summing (4.9) for $l = 1$ to k yields

(4.31)
$$\mathcal{E}(k) - \mathcal{E}(0) \le \alpha \sum_{l=1}^{k} \left((\phi_k - 2)\|\Delta\mathbf{x}_{l-1}\|^2 - \phi_k\|\Delta\mathbf{x}_l\|^2 \right) - \frac{2s}{r-1}\sum_{l=1}^{k}(f(\mathbf{x}_{l-1}) - f(\mathbf{x}^*))$$

$$\le \alpha \left(-\phi_k\|\Delta\mathbf{x}_k\|^2 - 2\sum_{l=2}^{k-1}\|\Delta\mathbf{x}_l\|^2 \right) - \frac{2s}{r-1}(f(\mathbf{x}_0) - f(\mathbf{x}^*)) \tag{4.32}$$

$$\le -\frac{2s}{r-1}(f(\mathbf{x}_0) - f(\mathbf{x}^*)). \tag{4.33}$$

∎

Theorem 3.2 tells that FISC-ODE has the $\mathcal{O}(t^{-2})$ convergence rate and the following theorem is a discretized analogue of Theorem 3.2.

THEOREM 4.1 (the $\mathcal{O}(k^{-2})$ convergence rate of FISC-PM). *Suppose that* $f(\mathbf{x}) = \psi(\mathbf{x}) + h(\mathbf{x})$, *where* $\psi \in \mathcal{F}_L$ *and* h *is convex. Assume that* f *has a unique minimizer* $\mathbf{x}^*$ *with* $\|\mathbf{x}^*\| < \infty$. *Let* $\{\mathbf{x}_k\}$ *be a sequence given by* (2.16) *and* (2.17). *The step size is fixed as* $0 < s \le 1/L$ *and* β_k, γ_k *are specified by* (2.4) *with* $l_k = k$. *Then, we have*

$$f(\mathbf{x}_k) - f(\mathbf{x}^*) \le \frac{(r-1)C_0}{2(k+r-2)^2 s} = \mathcal{O}(k^{-2}),$$

where

$$C_0 = \mathcal{E}(0) - \frac{2s}{r-1}(f(\mathbf{x}_0) - f(\mathbf{x}^*)) = 2\|\mathbf{x}_0 - \mathbf{x}^*\|^2 + (r-3)s\left(f(\mathbf{x}_0) - f(\mathbf{x}^*)\right).$$

Proof. By Lemma 6, the sequence of $\{\mathbf{x}_k\}$ given by FISC-PM satisfies

$$f(\mathbf{x}_k) - f(\mathbf{x}^*) \le \frac{r-1}{2(k+r-2)^2 s}\mathcal{E}(k) \le \frac{r-1}{2(k+r-2)^2 s}C_0 = \mathcal{O}(k^{-2}),$$

which completes the proof. ∎

Note that FISC-nes is FISC-PM with $h = 0$. Hence, we also prove the $\mathcal{O}(k^{-2})$ convergence rate of FISC-nes for smooth convex optimization problems.

5. Numerical experiments.

5.1. Sparse optimization. We compare FIRE, FISC, and other optimization solvers on the following problem:

$$\min_{\mathbf{x}\in\mathbb{R}^n} \frac{1}{2}\|A\mathbf{x} - b\|^2 + \lambda\|\mathbf{x}\|_1.$$

Here we have $\psi(\mathbf{x}) = \frac{1}{2}\|A\mathbf{x} - b\|^2$, $h(\mathbf{x}) = \lambda\|\mathbf{x}\|_1$, where $A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m, \lambda > 0$. The proximal mapping is computed as

$$\left(\text{prox}_h^s(\mathbf{x})\right)_i = \text{sign}(\mathbf{x}_i)\max\{|\mathbf{x}_i| - \lambda s, 0\}. \tag{5.1}$$

In our numerical experiment, λ varies from different test cases and it is around 8×10^{-3}.

5.1.1. Algorithm details and the implementation. We describe the implementation details of our method and of the state-of-the-art algorithms used in our numerical comparison. The solvers used for comparison include SNF [18], ASSN [31], FPC-AS [26], and SpaRSA [29]. We give an overview of the tested algorithms:

- SNF is a semismooth Newton type method which uses the filter strategy.
- SNF(aCG) is the SNF solver with an adaptive parameter strategy in the CG method for solving the Newton equation.
- ASSN is an adaptive semismooth Newton method.
- FPC-AS is a first-order method that uses a fixed-point iteration under Barzilai–Borwein (BB) steps [2] and the continuation strategy.
- SpaRSA, which resembles FPC-AS, is also a first-order method using BB steps and the continuation strategy.
- F-PG/F-PM/FS-PG(r)/FS-PM(r) is the FIRE/FISC algorithm using the proximal gradient (the proximal mapping) with the continuation strategy. The step size is obtained from the nonmonotone line search with the BB step as the initial guess. The number in the bracket is the parameter r for FISC-PG/FISC-PM.

The continuation strategy in F-PG/F-PM/FS-PG(r)/FS-PM(r) is the same as in [26]. Note that FISC-PM with $r = 3$ recovers FISTA. We take same parameters for ASSN, FPC-AS, SpaRSA, and SNF as in [18].

5.1.2. The numerical comparison. We use test problems from [18], which are constructed as follows. First, we randomly generate a sparse solution $\bar{\mathbf{x}} \in \mathbb{R}^n$ with k nonzero entries, where $n = 512^2 = 262144$ and $k = [n/40] = 5553$. The k different indices are uniformly chosen from $\{1, 2, ..., n\}$ and the magnitude of each nonzero element is set by $\bar{x}_i = c_1(i)10^{dc_2(i)/20}$, where $c_1(i)$ is randomly chosen from $\{-1, 1\}$ with probability 1/2, respectively, $c_2(i)$ is uniformly distributed in $[0, 1]$, and d is a dynamic range which can influence the efficiency of the solvers. Then we choose $m = n/8 = 32768$ random cosine measurements, i.e., $A\bar{\mathbf{x}} = (dct(\bar{\mathbf{x}}))_J$, where J contains m different indices randomly chosen from $\{1, 2, ..., n\}$ and dct is the discrete cosine transform. Finally, we construct the input data by $b = A\bar{\mathbf{x}} + w$, where w is an isotropic Gaussian noise with a standard deviation $\bar{\sigma} = 0.1$.

To compare fairly, we set a uniform stopping criterion. For a certain tolerance ϵ, we obtain a solution $\mathbf{x}_{newt}$ using ASSN [31] such that $\|sG_s(\mathbf{x}_{newt})\| \leq \epsilon$. Then, we terminate all methods by the relative criterion

$$\frac{f\left(\mathbf{x}^k\right) - f\left(\mathbf{x}^*\right)}{\max\left\{\left|f\left(\mathbf{x}^*\right)\right|, 1\right\}} \leq \frac{f\left(\mathbf{x}_{newt}\right) - f\left(\mathbf{x}^*\right)}{\max\left\{\left|f\left(\mathbf{x}^*\right)\right|, 1\right\}},$$

where $f(\mathbf{x})$ is the objective function and $\mathbf{x}^*$ is a highly accurate solution using ASSN [31] under the criterion $\|sG_s(\mathbf{x}^*)\| \leq 10^{-13}$.

We solve the test problems under different tolerances $\epsilon \in \{10^{-0}, 10^{-1}, 10^{-2}, 10^{-4}, 10^{-6}\}$ and dynamic ranges $d \in \{20, 40, 60, 80\}$. Since the evaluations of dct (in computing $A\mathbf{x}$ and $A^T\mathbf{x}$) dominate the overall computation we mainly use the total numbers of A-calls and A^T-calls N_A to compare the efficiency of different solvers. Tables 5.1–5.4 show the average numbers of N_A and CPU time over 10 independent trials.

From the numerical results, with the increase of the dynamic range, FS-PG(5) is competitive with ASSN or even outperforms ASSN in terms of both cpu time and N_A. If only a low precision is required, i.e., $\epsilon = 10^0$, FPC-AS has the smallest N_A with dynamic ranges 40dB, 60dB, and 80dB. With a relative low precision of ϵ, F-PM achieves better performance than FS-PG(5). Although in one iteration F-PM has to calculate the proximal gradient twice, F-PM performs much better than F-PG. In general, FISC with $r = 5$ has better performance than FISC with $r = 3$. These observations indicate the strength of SDC in general.

TABLE 5.1
Total number of A-calls and A^T-calls N_A and CPU time (in seconds) averaged over 10 *independent runs with dynamic range* $20dB$.

Method	$\epsilon : 10^0$		$\epsilon : 10^{-1}$		$\epsilon : 10^{-2}$		$\epsilon : 10^{-4}$		$\epsilon : 10^{-6}$	
	Time	N_A	Time	N_A	Time	N_A	Time	N_A	Time	N_A
SNF	1.09	84.6	2.63	205.0	3.20	254.2	3.85	307.0	4.59	373.0
SNF(aCG)	1.11	84.6	2.62	205.0	3.24	254.2	4.13	331.2	6.62	486.2
ASSN	1.13	89.8	1.82	145.0	2.10	**173.0**	**2.97**	**246.4**	**3.55**	**298.2**
FPC-AS	1.45	109.8	5.08	366.0	6.88	510.4	9.56	719.4	9.90	740.8
SpaRSA	4.92	517.2	4.84	519.2	5.12	539.8	5.86	627.0	6.61	705.8
F-PG	2.14	190.4	3.21	291.2	4.25	376.8	6.79	600.8	9.05	801.8
FS-PG(3)	0.81	71.2	1.34	**119.4**	**1.93**	175.6	3.24	283.8	4.48	394.8
FS-PG(5)	**0.70**	**64.4**	**1.32**	121.2	2.07	182.0	3.24	286.6	4.39	390.2
F-PM	0.95	81.8	1.54	140.0	2.11	180.4	3.91	338.8	5.14	464.2
FS-PM(3)	1.12	97.0	1.97	168.0	3.51	298.6	6.71	596.0	9.39	817.0
FS-PM(5)	0.98	87.4	1.68	141.4	2.65	227.0	6.36	560.2	8.08	702.2

TABLE 5.2
Total number of A-calls and A^T-calls N_A and CPU time (in seconds) averaged over 10 *independent runs with dynamic range* $40dB$.

Method	$\epsilon : 10^0$		$\epsilon : 10^{-1}$		$\epsilon : 10^{-2}$		$\epsilon : 10^{-4}$		$\epsilon : 10^{-6}$	
	Time	N_A	Time	N_A	Time	N_A	Time	N_A	Time	N_A
SNF	2.06	158.2	5.01	380.8	6.19	483.2	6.69	525.0	7.16	566.8
SNF(aCG)	2.08	158.2	4.97	380.8	6.16	483.2	7.07	553.6	7.30	580.0
ASSN	2.28	182.2	3.53	285.4	4.10	338.6	4.97	**407.0**	**5.56**	**459.2**
FPC-AS	2.12	158.0	5.34	399.2	7.72	578.4	9.62	720.2	10.41	774.8
SpaRSA	5.05	523.4	5.07	530.0	5.56	588.2	6.38	671.6	7.28	755.8
F-PG	4.28	378.0	5.76	522.4	7.28	642.8	9.28	813.6	11.05	990.0
FS-PG(3)	1.71	153.6	3.05	276.4	3.94	354.6	4.89	439.6	6.37	567.2
FS-PG(5)	**1.62**	**143.6**	**2.72**	**245.6**	**3.46**	**317.6**	**4.41**	415.6	5.68	518.0
F-PM	2.02	171.2	2.68	244.0	3.94	347.4	5.34	480.8	7.09	626.2
FS-PM(3)	2.11	184.2	3.14	279.8	4.68	424.0	7.29	648.2	10.11	903.4
FS-PM(5)	2.17	191.2	3.50	308.8	4.54	401.4	6.07	537.0	8.25	716.8

TABLE 5.3
Total number of A-calls and A^T-calls N_A and CPU time (in seconds) averaged over 10 *independent runs with dynamic range* $60dB$.

Method	$\epsilon : 10^0$		$\epsilon : 10^{-1}$		$\epsilon : 10^{-2}$		$\epsilon : 10^{-4}$		$\epsilon : 10^{-6}$	
	Time	N_A	Time	N_A	Time	N_A	Time	N_A	Time	N_A
SNF	5.12	391.8	8.28	648.8	9.86	777.6	10.44	828.2	11.13	881.0
SNF(aCG)	5.05	391.8	8.32	648.8	9.89	777.6	10.83	861.2	11.37	903.2
ASSN	3.60	295.4	5.01	416.4	5.95	492.0	6.97	**582.4**	7.66	**642.4**
FPC-AS	**3.14**	**232.2**	8.89	644.0	11.61	844.4	13.80	1004.4	14.08	1031.2
SpaRSA	5.48	561.2	5.69	598.2	6.57	683.2	7.70	797.8	8.62	900.6
F-PG	7.07	638.6	8.77	780.8	10.35	937.2	13.05	1157.2	14.85	1338.0
FS-PG(3)	3.53	328.6	4.58	422.0	5.60	506.0	6.83	619.8	7.96	714.6
FS-PG(5)	3.49	319.0	4.58	428.6	5.72	520.6	**6.71**	612.8	**7.58**	695.0
F-PM	3.53	310.4	**4.14**	**374.0**	**5.43**	**485.8**	7.98	720.2	9.65	868.6
FS-PM(3)	3.76	342.0	4.74	429.8	6.50	584.8	10.52	950.2	13.32	1201.2
FS-PM(5)	3.48	307.6	4.19	383.2	5.53	502.4	8.01	703.4	9.33	848.8

TABLE 5.4
Total number of A-calls and A^T-calls N_A and CPU time (in seconds) averaged over 10 *independent runs with dynamic range* $80dB$.

Method	$\epsilon : 10^0$		$\epsilon : 10^{-1}$		$\epsilon : 10^{-2}$		$\epsilon : 10^{-4}$		$\epsilon : 10^{-6}$	
	Time	N_A	Time	N_A	Time	N_A	Time	N_A	Time	N_A
SNF	7.65	591.0	10.87	841.6	12.49	978.6	13.08	1024.8	15.89	1227.6
SNF(aCG)	7.58	591.0	10.78	841.6	12.44	978.6	13.30	1042.2	13.99	1105.8
ASSN	5.96	482.8	7.47	601.0	8.39	690.6	9.52	780.6	10.32	852.6
FPC-AS	**4.28**	**321.4**	8.28	611.0	10.61	788.0	11.85	883.2	12.13	902.0
SpaRSA	5.18	543.2	6.26	665.4	7.35	763.0	8.26	871.8	8.98	942.0
F-PG	7.18	642.8	8.90	792.8	10.35	951.0	12.47	1134.8	13.50	1231.6
FS-PG(3)	4.85	444.8	6.09	555.4	7.01	649.2	7.76	727.0	8.65	789.2
FS-PG(5)	4.30	407.2	5.72	521.6	6.77	625.8	**7.64**	**702.0**	**8.15**	**753.2**
F-PM	4.17	388.8	**5.26**	**463.2**	**6.55**	583.2	8.14	729.2	9.06	814.6
FS-PM(3)	6.00	533.4	6.87	635.4	8.41	748.4	13.08	1162.8	15.04	1348.4
FS-PM(5)	4.99	436.4	5.75	525.0	7.08	639.8	9.51	860.0	10.93	987.0

5.2. Logistic regression. We consider the ℓ_1-logistic regression problem

$$\min_{\mathbf{x}=(\hat{\mathbf{x}},y)\in\mathbb{R}^{n+1}} \frac{1}{N}\sum_{i=1}^{N}\log(1+\exp(-b_i(\langle \mathbf{a}_i,\hat{\mathbf{x}}\rangle + y))) + \lambda\|\mathbf{x}\|_1, \tag{5.2}$$

where data pairs $(\mathbf{a}_i, b_i) \in \mathbb{R}^n \times \{-1, 1\}$ correspond to a given data set. The regularization parameter $\lambda > 0$ controls the level of sparsity of a solution to (5.2). In our numerical experiments, λ is set to be 0.001.

5.2.1. Algorithm details and the implementation. The solvers include prox-SVRG [30], Adagrad [8], and SGD. We give an overview of the tested methods:

- *prox-SVRG* stands for a variance reduced stochastic proximal gradient method. Similar to [19], we substitute the basic variance reduction technique proposed in [30] with the minibatch version (2.20).
- *Adagrad* is a stochastic proximal gradient method with a specific strategy for choosing adaptive step sizes. We use the minibatch gradient (2.19) as the first-order oracle in our implementation.
- *SGD* is a stochastic proximal gradient method. The minibatch gradient (2.19) is used as the first-order oracle in our implementation.
- *sF-PG/sFS-PG(r)* stands for the stochastic version of FIRE/FISC using the proximal gradient. The stochastic oracle (2.19) is used. In FISC, we take $r = 3$ and $r = 7$.
- *sFVR-PG/sFSVR-PG(r)* stands for the stochastic version of FIRE/FISC using the proximal gradient. The variance reduced stochastic oracle (2.20) is used. In FISC, we take $r = 3$ and $r = 7$.

For all solvers, the sample size is fixed to be $|\mathcal{S}_k| = \lfloor 0.01N \rfloor$. The proximal operator of the ℓ_1-norm is given in (5.1). In SVRG, we set $m = 200$ in (2.20); in sFVR-PG/sFSVR-PG, we set $m = 20$ in (2.20). Here we intentionally set a larger m in SVRG because it generates a higher precision solution.

5.2.2. The numerical comparison. The tested data sets obtained from libsvm [5] in our numerical comparison are summarized in Table 5.5. Except for two large data sets tfidf and log1p, we linearly scale the entry of the data-matrix $\mathbf{A} = (\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_N)$ to $[0, 1]$. Then, we add a row of ones into the data-matrix $\mathbf{A}$ as coefficients for the bias term in our linear classifier. The data sets for multiclass classification have been manually divided into two types of features. For instance, the MNIST data set is used to classify even and odd digits.

TABLE 5.5
Information of the data sets in ℓ_1-logistic regression. Here "sparsity" represents the proportion of nonzero entries in $\mathbf{A}$.

Data set	Data points N	Variables n	sparsity
rcv1	20,242	47,236	0.16%
CINA	16,033	132	29.56%
MNIST	60,000	784	19.12%
gisette	6,000	5,000	12.97%
mushroom	8,124	112	18.75%
synthetic	10,000	50	22.12%
tfidf	16,087	150,360	0.83%
log1p	16,087	4,272,227	0.14%

TABLE 5.6
Initial step sizes.

Solver	prox-SVRG	Adagrad	SGD	sF-PG	sFS-PG	sFVR-PG	sFSVR-PG
rcv1	8	2^{-4}	32	32	32	8	16
CINA	2	2^{-3}	8	8	8	2	2
MNIST	0.5	2^{-5}	1	1	1	0.5	0.5
gisette	0.5	2^{-5}	2	1	2	0.5	0.5
mushroom	128	8	8	128	128	128	128
synthetic	2	0.125	4	4	4	2	2
tfidf	2	0.25	1	0.25	0.5	0.25	0.25
log1p	32	0.5	16	16	16	32	32

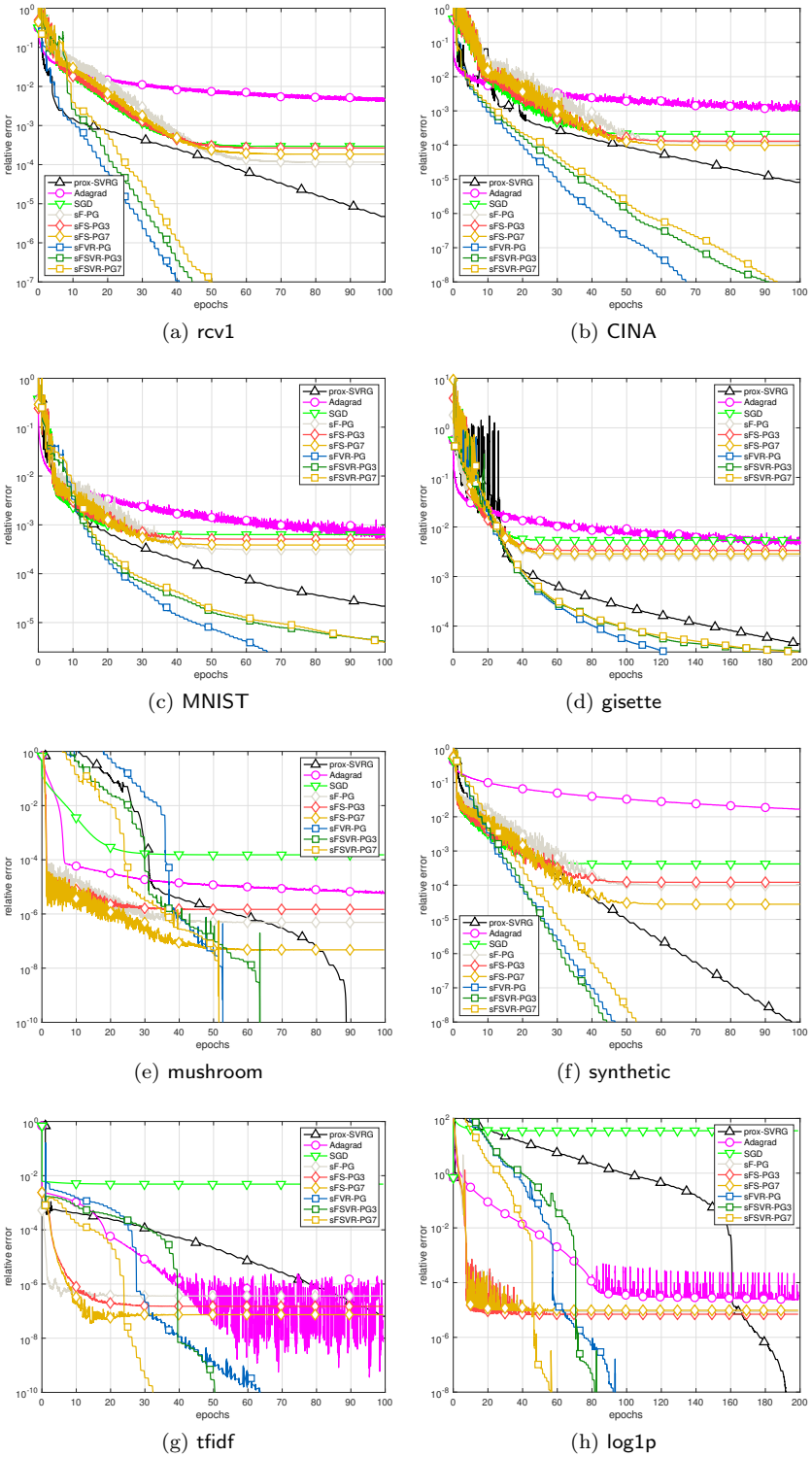
The initial step size varies for different tested data sets and it determines the performance of solvers. Hence, we chose the initial step size from set $\{2^i | i \in \{-7, -6, \ldots, 7\}\}$. For each data set, we ran the algorithms with these different parameters and selected a parameter that ensured the best overall performance. Table 5.6 gives the initial step size over these data sets. For SGD, sF(S)-PG, and sF(S)VR-PG, we use an exponentially decaying step size. Namely, we decrease the step size by multiplying 0.85 in each epoch. For all methods, we choose $\mathbf{x}_0 = 0$ as the initial point.

We next show the performance of all methods. The change of the *relative error* $(f(\mathbf{x}) - f(\mathbf{x}^*))/(\max\{1, |f(\mathbf{x}^*)|\})$ is reported with respect to epoch. Here one epoch means going through the entire dataset for one time and it involves several iterations since the sample size is fixed to be $|\mathcal{S}_k| = \lfloor 0.01N \rfloor$. The point x^* is a reference solution of problem (5.2) generated by S2N-D in [19] with a stopping criterion $\|\mathbf{x}_k - \mathbf{x}_{k-1}\| < 10^{-12}$. The numerical results are plotted in Figure 5.1. We average the results over 10 independent runs except that only one run is used for log1p because the execution time is too long.

In Figure 5.1, we can roughly split these stochastic methods into two categories: with and without variance reduction techniques. The first category includes sFVR-PG, sFSVR-PG, and prox-SVRG, while the second category consists of sF-PG, sFS-PG, SGD, and Adagrad. For methods in the first category, we observe that sFVR-PG and sFSVR-PG defeat all other methods, especially in epoch. sFSVR-PG(7) has competitive performance compared to sFSVR-PG and sFSVR-PG(3). The variance reduction technique seems to be especially well-suited for stochastic FIRE/FISC. On log1p, SVRG decreases slowly in the early stage of iterations but converges rapidly when the iterates are close to an optimal solution.

On most test cases, sFS-PG(7) achieves the best performance both with respect to relative error and epoch among other methods, when variance reduction techniques are not used. Our observation indicates that methods with SDC, i.e., sF-PG and sFS-PG, outperform SGD and Adagrad. On large data sets, like tfidf and log1p, SGD converges to a solution with low precision. Adagrad experiences oscillation after 100 epochs. Although sF-PG and sFS-PG experience oscillation at first, they finally converge to a precise solution.

In general, sFS-PG(7) is better than sFS-PG(3) and it has similar performance to sF-PG. While sFVR-PG and sFSVR-PG(3) slightly outperform sFSVR-PG(7) in some test cases, sFSVR-PG(7) can lead to a more accurate solution on data sets such as mushroom, tfidf, and log1p within some given number of epochs. Overall, our numerical results indicate that SDC, especially combined with variance reduction techniques, is very promising.



(a) rcv1 (b) CINA

(c) MNIST (d) gisette

(e) mushroom (f) synthetic

(g) tfidf (h) log1p

FIG. 5.1. *Change of the relative error with respect to the epochs for solving the ℓ_1-logistic regression problem (averaged over 10 independent runs, except for* log1p*).*

TABLE 5.7
*The number of parameters of DenseNet*12*/ResNet*34 *on CIFAR*-10*/CIFAR*-100.

	DenseNet121	ResNet34
CIFAR-10	6,956,298	21,282,122
CIFAR-100	7,048,548	21,328,292

TABLE 5.8
The initial learning rate.

	CIFAR-10	CIFAR-100
MSGD	0.1	0.1
Adam	0.001	0.001
FIRE	0.01	0.1
FISC	0.01	0.1

5.3. Deep learning. The optimization problem in deep learning is

$$\min_{\mathbf{x}\in\mathbb{R}^n} \frac{1}{N}\sum_{i=1}^{N} l\left(f\left(\mathbf{a}^{(i)},\mathbf{x}\right),b^{(i)}\right)+\lambda\|\mathbf{x}\|_2^2,$$

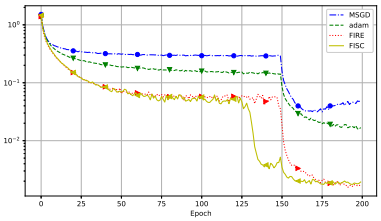
where $\mathbf{x}$ denotes the parameters for training, data pairs $\{(\mathbf{a}^{(i)},b^{(i)})\}$ correspond to a given data set, $f(\cdot,\mathbf{x})$ represents the function determined by the neural network architecture, $l(\cdot,\cdot)$ denotes the loss function, and λ is the coefficient of weight decay (ℓ_2-regularization).

We evaluate our proposed algorithm on deep learning for image classification tasks using the benchmark data sets: CIFAR-10 and CIFAR-100 [14]. CIFAR-10 is a database of images from 10 classes and CIFAR-100 consists of images drawn from 100 classes. Both of them consist of 50,000 training images and 10,000 test images. We normalize the data using the channel means and standard deviations for preprocessing. The neural network architectures include DenseNet121 [11] and ResNet34 [10]. The number of parameters is listed in Table 5.7.

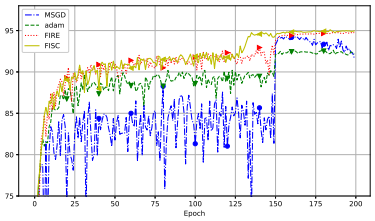
The implemented algorithms include SGD with momentum (MSGD), Adam [13], FIRE, and FISC with $r = 7$. The initial learning rate for different methods is given in Table 5.8. On CIFAR-10, we train the network using a batch size 128 for 200 epochs. The coefficient λ is 5×10^{-4}. The learning rate is decreased 10 times at epoch 150, which is the same as [17]. On CIFAR-100, we train the network using a batch size 64 for 300 epochs and λ is 1×10^{-4}. The learning rate is multiplied by 0.1 at epoch 150 and epoch 225, which is the same as [11]. For both data sets, the momentum factor is 0.9 in MSGD, FIRE, and FISC; (β_1,β_2) in Adam are $(0.9, 0.999)$ on DenseNet and $(0.99, 0.999)$ on ResNet; ϵ in Adam is 10^{-8}.

Figures 5.2 and 5.3 show that on CIFAR-10, FISC and FIRE have better performance than MSGD and Adam from the very beginning, especially in training loss. On CIFAR-10 with DenseNet, the test accuracy of FISC approaches 95% around epoch 130. On CIFAR-100 with DenseNet, FISC and FIRE outperform MSGD and Adam in test accuracy. This further illustrates the strength of SDC.

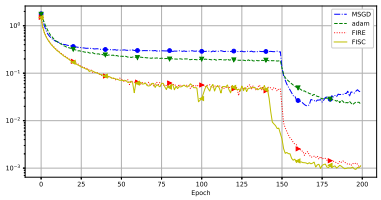
6. Conclusion. In this paper, we propose a family of first-order methods with SDC. The restarting condition is the foundation for the global convergence of methods with SDC. From an ODE perspective, we construct the FISC-ODE with an $\mathcal{O}(t^{-2})$ convergence rate. FISC-PG shows excellent performance in numerical experiments, while FISC-PM has a provable $\mathcal{O}(k^{-2})$ convergence rate. Numerical experiments indicate that our algorithmic framework with SDC is competitive and promising.



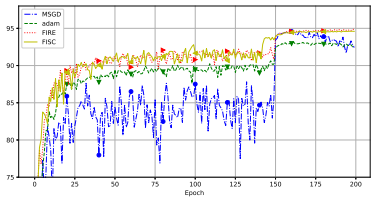
(a) Training Loss, DenseNet121



(b) Test Accuracy, DenseNet121

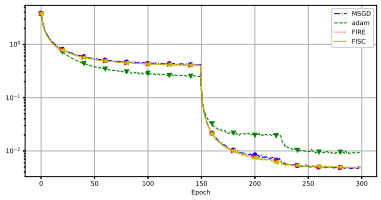


(c) Training Loss, ResNet34

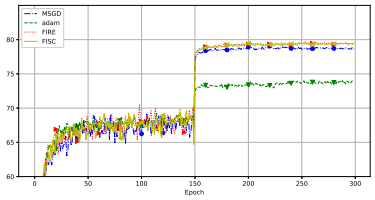


(d) Test Accuracy, ResNet34

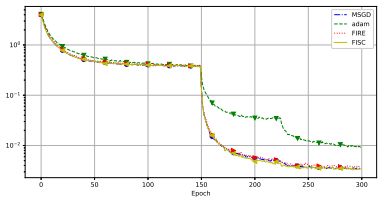
FIG. 5.2. *Numerical results on CIFAR*-10. *Top: DenseNet*121*; Bottom: ResNet*34.



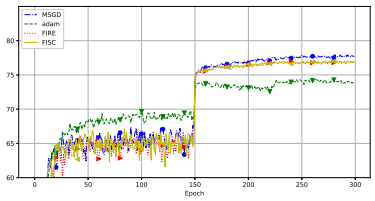
(a) Training Loss, DenseNet121



(b) Test Accuracy, DenseNet121



(c) Training Loss, ResNet34



(d) Test Accuracy, ResNet34

FIG. 5.3. *Numerical results on CIFAR*-100. *Top: DenseNet*121*; Bottom: ResNet*34.

Acknowledgments. The authors are grateful to Professor Andrea Walther and two anonymous referees for their valuable comments and suggestions. Zaiwen Wen would like to thank Lin Lin and Chao Yang for the kind introduction to and discussion on the FIRE method. part by the NSFC grants 11421101 and 11831002, and by the National Basic Research Project under the grant 2015CB856002.

REFERENCES

[1] H. Attouch, Z. Chbani, J. Fadili, and H. Riahi, *First-order optimization algorithms via inertial systems with hessian driven damping*, Math. Program., 2020, https://doi.org/10.1007/s10107-020-01591-1.

[2] J. Barzilai and J. M. Borwein, *Two-point step size gradient methods*, IMA J. Numer. Anal., 8 (1988), pp. 141–148.

[3] A. Beck and M. Teboulle, *A fast iterative shrinkage-thresholding algorithm for linear inverse problems*, SIAM J. Imaging Sci., 2 (2009), pp. 183–202.

[4] E. Bitzek, P. Koskinen, F. Gähler, M. Moseler, and P. Gumbsch, *Structural relaxation made simple*, Phys. Rev. Lett., 97 (2006).

[5] C.-C. Chang and C.-J. Lin, *LIBSVM: A library for support vector machines*, ACM Trans. Intell. Syst. Technol., 2 (2011), 27.

[6] Y. Dai, *Nonlinear Conjugate Gradient Methods*, Shanghai Science and Technology, Shanghai, 2000.

[7] A. Defazio, F. Bach, and S. Lacoste-Julien, *SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives*, in Proceedings of Neural Information Processing Systems, 2014, pp. 1646–1654.

[8] J. Duchi, E. Hazan, and Y. Singer, *Adaptive subgradient methods for online learning and stochastic optimization*, J. Mach. Learn. Res., 12 (2011), pp. 2121–2159.

[9] E. Ghadimi, H. R. Feyzmahdavian, and M. Johansson, *Global convergence of the heavy-ball method for convex optimization*, in Proceedings of the European Control Conference (ECC), IEEE, 2015, pp. 310–315.

[10] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.

[11] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, *Densely connected convolutional networks*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 4700–4708.

[12] R. Johnson and T. Zhang, *Accelerating stochastic gradient descent using predictive variance reduction*, in Proceedings of the Advances in Neural Information Processing Systems, 2013, pp. 315–323.

[13] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, in Proceedings of the International Conference on Learning Representations 2015.

[14] A. Krizhevsky, *Learning Multiple Layers of Features from Tiny Images*, Master's thesis, Department of Computer Science, University of Toronto, 2009.

[15] L. Lessard, B. Recht, and A. Packard, *Analysis and design of optimization algorithms via integral quadratic constraints*, SIAM J. Optim., 26 (2016), pp. 57–95.

[16] D. C. Liu and J. Nocedal, *On the limited memory BFGS method for large scale optimization*, Math. Program., 45 (1989), pp. 503–528.

[17] L. Luo, Y. Xiong, Y. Liu, and X. Sun, *Adaptive gradient methods with dynamic bound of learning rate*, in Proceedings of the International Conference on Learning Representations, 2018.

[18] A. Milzarek and M. Ulbrich, *A semismooth Newton method with multidimensional filter globalization for l_1-optimization*, SIAM J. Optim., 24 (2014), pp. 298–333.

[19] A. Milzarek, X. Xiao, S. Cen, Z. Wen, and M. Ulbrich, *A stochastic semismooth Newton method for nonsmooth nonconvex optimization*, SIAM J. Optim., 29 (2019), pp. 2916–2948.

[20] Y. Nesterov, *A method of solving a convex programming problem with convergence rate $O(1/k^2)$*, Soviet Math. Dokl., 27 (1983), pp. 372–376.

[21] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*, Appl. Optim. 87, Springer, New York, 2013.

[22] B. O'Donoghue and E. J. Candés, *Adaptive restart for accelerated gradient schemes*, Found. Comput. Math., 15 (2015), pp. 715–732.

[23] B. T. Polyak, *Introduction to Optimization*, Optimization Software, New York, 1987.

[24] M. Schmidt, N. LeRoux, and F. Bach, *Minimizing Finite Sums with the Stochastic Average Gradient*, Technical report, INRIA, 2013.

[25] W. Su, S. Boyd, and E. J. Candès, *A differential equation for modeling Nesterov's accelerated gradient method: Theory and insights*, J. Mach. Learn. Res., 17 (2016).

[26] Z. Wen, W. Yin, W. Goldfarb, and D. Zhang, *A fast algorithm for sparse reconstruction based on shrinkage, subspace optimization, and continuation*, SIAM J. Sci. Comput., 32 (2010), pp. 1832–1857.

[27] A. Wibisono, A. C. Wilson, and M. I. Jordan, *A variational perspective on accelerated methods in optimization*, Proc. Natl. Acad. Sci. USA, 113 (2016), pp. E7351–E7358.

[28] A. C. Wilson, B. Recht, and M. I. Jordan, *A Lyapunov Analysis of Momentum Methods in Optimization*, https://arxiv.org/abs/1611.02635, 2016.
[29] S. J. Wright, R. D. Nowak, and M. A. Figueiredo, *Sparse reconstruction by separable approximation*, IEEE Trans. Signal Process., 57 (2009), pp. 2479–2493.
[30] L. Xiao and T. Zhang, *A proximal stochastic gradient method with progressive variance reduction*, SIAM J. Optim., 24 (2014), pp. 2057–2075.
[31] X. Xiao, Y. Li, Z. Wen, and L. Zhang, *A regularized semi-smooth Newton method with projection steps for composite convex programs*, J. Sci. Comput., 76 (2018), pp. 364–389.
[32] H. Zhang and W. W. Hager, *A nonmonotone line search technique and its application to unconstrained optimization*, SIAM J. Optim., 14 (2004), pp. 1043–1056.
[33] J. Zhang, A. Mokhtari, S. Sra, and A. Jadbabaie, *Direct Runge-Kutta discretization achieves acceleration*, in Proceedings of the 32nd International Conference on Neural Informational Processing Systems, 2018, pp. 3900–3909.