A Unified Framework to Understand Decentralized and Federated Optimization Algorithms: A Multi-Rate Feedback Control Perspective

Xinwei Zhang, Mingyi Hong, Nicola Elia
Department of Electric and Computer Engineering
Minnesota University
`zhan6234, mhong, nelia@umn.edu`

Abstract

We propose a unified framework to analyze and design distributed optimization algorithms. Through the lens of multi-rate feedback control, we show that a wide class of distributed algorithms, including popular decentralized/federated schemes such as decentralized gradient descent, gradient tracking, and federated averaging, among others, can be viewed as discretizing a continuous time feedback control system, but with different discretization patterns and/or multiple sampling rates. This key observation not only allows us to develop a generic framework to analyze the convergence of the entire algorithm class, more importantly, it leads to a new way of designing new distributed algorithms. We develop the theory behind our framework, and provide an example to highlight how the framework can be used to analyze and extend the well-known gradient tracking algorithm.

1 Introduction

Distributed computation has played an important role in machine learning, partly due to the dramatically increased size of the models and the datasets; see [1, 2] for a few recent surveys. Heterogeneous computational and communication resources in the distributed system create a number of different scenarios in distributed learning. For example, in a *decentralized optimization (DO)* setting, the communication and computation resources are equally important, so the algorithms alternatingly perform communication and communication steps [3, 4, 5, 6, 7]; In the *Federated Learning (FL)* setting, the communication is the bottleneck of the system, so the algorithms typically perform multiple local updates before one communication step [8, 9, 10, 11]; Additionally, in order to identify the *the optimal* decentralized algorithms that utilize the *minimum* computation and communication rounds, it is typically required to perform multiple communication steps before one local update [12, 13].

However, there are a few looming concerns and challenges over the proliferation of these algorithms. First, for some relatively hot problems, there are simply *too many algorithms* available, so much so that it becomes difficult to track all the technical details. Is it possible to establish some general guidelines to understand the fundamental relationships between algorithms that take similar forms, or provide similar features/functionalities? Second, much of the recent research on this topic appears to be *increasingly focused* on a specific setting (e.g., those that mentioned in the previous section). However, an algorithm developed for FL may have already been rigorously developed, analyzed, and tested for the DO setting, with only minor differences. Developing an algorithm and its corresponding analysis takes significant time and effort, therefore it is desirable to have some mechanisms in place to reduce the possibility of "reinventing the wheel". We argue that there is a strong demand of a framework for distributed optimization, which can help researchers and practitioners to **simplify** their understanding about algorithm behaviors, **predict** performance, and **streamline** algorithm design.

1st NeurIPS Workshop on New Frontiers in Federated Learning (NFFL 2021), Virtual Meeting.

Our main contribution is to build such a unified framework for distributed algorithms, using tools from multi-rate feedback control systems. Specifically, we first show that, a special continuous-time feedback control system is well-suited to capture a number of key properties of distributed algorithms. We then show that when such a continuous-time system is discretized appropriately (in which different parts of the system are discretized using different rates, hence the name "multi-rate" system), it recovers a wide range of decentralized/federated algorithms. Finally, we provide a generic convergence result that covers different feedback schemes as well as discretization patterns. The major benefits of our proposed framework can be summarized as:

1) Using our framework, we can establish the connection between different subclasses of distributed algorithms that are developed for different settings; in some sense, they can be viewed as applying different discretization schemes to certain continuous-time control system;

2) Our framework helps predict the algorithm performance, and facilitates algorithm design – as long as the continuous-time control system and the desired discretization pattern are identified, our framework readily provides the various system parameters that are needed to ensure algorithm convergence (an example is provided in the appendix to showcase how this can be done).

Note that there are many existing works which analyze optimization algorithms using control theory, but they mainly focus on some very special class of algorithms. For examples, [14, 15, 16] study a restrictive class of simple convex optimization algorithms; the paper [17, 18, 19] investigates the acceleration approaches for centralized problems in discrete time; [15, 19] focus on the continuous-time system and ignore the impact of the discretization to these algorithms; [20, 21, 22] investigate the connection between continuous-time system and discretized gradient descent algorithm, but their approaches and analyses do not generalize to federated/decentralized algorithms. It is also important to note that, to our knowledge, none of the above referred works provide any insights about relationship between difference classes of distributed algorithms (i.e., between DO and FL), nor do they facilitate the design of new algorithms.

2 Continuous-time System

In this section, we provide a general description of the continuous-time multi-agent feedback control system. We start by giving a general system structure and discuss the property of each controller and how the controllers are related to the discrete-time optimization algorithms. Then we provide the convergence properties of the system.

2.1 System Description

The Decentralized Optimization Problem. Consider a distributed system with N agents connected by a strongly connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, each optimizes a smooth and possibility non-convex local function $f_i(x)$. The global optimization problem is formulated as [15]

$$\min_{\mathbf{x}\in\mathbb{R}^{Nd_x}} f(\mathbf{x}) := \frac{1}{N}\sum_{i=1}^{N} f_i(x_i) \qquad \text{s.t.} \;\; (A\otimes I)\cdot\mathbf{x} = 0, \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^{N\times d_x}$ stacks N local variables $\mathbf{x} := [x_1; \ldots; x_N]$; $x_i \in \mathbb{R}^{d_x}$, $\forall\, i \in [N]$; $\otimes$ denotes the Kronecocker product; the incidence matrix A contains the graph connectivity pattern with the following definition: if edge $e(i,j) \in \mathcal{E}$ connects vertex i and j with $i > j$, then $A_{ei} = 1$, $A_{ej} = -1$ and $A_{ek} = 0$, $\forall k \neq i, j$. Let us use $\mathcal{N}_i \subset [N]$ denote the neighbors for agent i. For simplicity of notation, the Kronecocker products are ignored in the latter analyses.

The Continuous-Time Double-Feedback System. To optimize problem (1), our approach is to design a continuous-time feedback control system, in such a way that the system enters its stable state if and only if the state variables of the system precisely correspond to a first-order stationary solution of (1). Towards this end, let us use $\mathbf{x} \in \mathbb{R}^{N\times d_x}$ to denote the main state variable of the system. Let us introduce two feedback loops, referred to as the *global consensus feedback loop* (GCFL) and *local computation feedback loop* (LCFL), where the former incorporates the dynamics from multi-agent interactions, while the latter helps better stabilize the system. More specifically, these loops can be specified as below:

• **(The GCFL).** Let us define an auxiliary state variable $\mathbf{v} := [v_1; \ldots; v_N] \in \mathbb{R}^{Nd_v}$, with $v_i \in \mathbb{R}^{d_v}$, $\forall\, i$, and define $\mathbf{y} = [\mathbf{x}; \mathbf{v}] \in \mathbb{R}^{N(d_x+d_v)}$; define a feedback controller $G_g(\cdot; A) : \mathbb{R}^{N(d_x+d_v)} \to \mathbb{R}^{N(d_x+d_v)}$. Then the GCFL will use the controller $G_g(\cdot; A)$ to operate on $\mathbf{y}$, to ensure that the agents remain coordinated, and their local control variables remain close to consensus;

• **(The LCFL).** Let us define an auxiliary state variable $\mathbf{z} := [z_1; \dots; z_N] \in \mathbb{R}^{Nd_z}$, with $z_i \in \mathbb{R}^{d_z}$, $\forall\, i$; define a set of feedback controller $G_\ell(\cdot; f_i) : \mathbb{R}^{d_x+d_v+d_z} \to \mathbb{R}^{d_x+d_v+d_z}$, one for each agent i. Then each agent will utilize LCFL to operate on its local state variables x_i, z_i and v_i, to ensure that without interacting with the neighboring agents, its local system can be stabilized.

Please see Figure 1 for an illustration of the continuous-time system mentioned above. It is clear that the system can be described by the following dynamics:

$$\begin{aligned}\dot{\mathbf{v}}(t) &= -\eta_g(t) \cdot u_{g,v}(t) - \eta_\ell(t) \cdot u_{\ell,v}(t)\\ \dot{\mathbf{x}}(t) &= -\eta_g(t) \cdot u_{g,x}(t) - \eta_\ell(t) \cdot u_{\ell,x}(t) \qquad (2)\\ \dot{\mathbf{z}}(t) &= -\eta_\ell(t) \cdot u_{\ell,z}(t).\end{aligned}$$

Next, we discuss in detail how different controllers work.

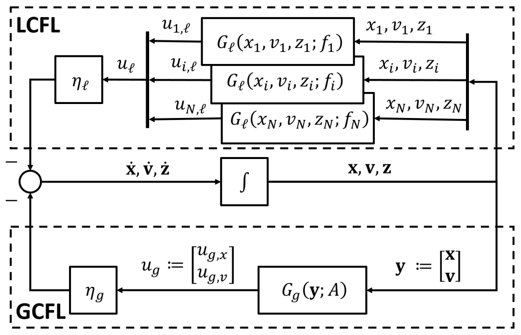

LCFL
$u_{1,\ell}$
$G_\ell(x_1, v_1, z_1; f_1)$
x_1, v_1, z_1
η_ℓ
u_ℓ
$u_{i,\ell}$
$G_\ell(x_i, v_i, z_i; f_i)$
x_i, v_i, z_i
$u_{N,\ell}$
$G_\ell(x_N, v_N, z_N; f_N)$
x_N, v_N, z_N
$-$
$\dot{\mathbf{x}}, \dot{\mathbf{v}}, \dot{\mathbf{z}}$
$\int$
$\mathbf{x}, \mathbf{v}, \mathbf{z}$
$-$
$u_g := \begin{bmatrix} u_{g,x} \\ u_{g,v} \end{bmatrix}$
$\mathbf{y} := \begin{bmatrix} \mathbf{x} \\ \mathbf{v} \end{bmatrix}$
η_g
$G_g(\mathbf{y}; A)$
GCFL

Figure 1: The proposed continuous-time double-feedback control system for modeling the decentralized optimization problem (1).

2.2 Global Consensus Feedback Loop

The GCFL performs inter-agent communication based on the incidence matrix A, and it controls the consensus of the global variable $\mathbf{y} := [\mathbf{x}; \mathbf{v}]$. Specifically, at time t, the controller $G_g(\mathbf{y}(t); A(t))$ can be further decomposed into two sub-controllers $G_{g,x}(\mathbf{y}(t); A(t))$ and $G_{g,v}(\mathbf{y}(t); A(t))$, where they produce control signals for $\mathbf{x}$ and $\mathbf{v}$, respectively, as follows:

$$u_g(t) = [u_{g,x}(t); u_{g,v}(t)], \ \text{ where } \ u_{g,x}(t) := G_{g,x}(\mathbf{y}(t); A(t)), \ u_{g,v}(t) := G_{g,v}(\mathbf{y}(t); A(t)).$$

After multiplied by the control gain $\eta_g(t) > 0$, the resulting signal will be combined with the output of the LCFL, and be fed back to local controllers. Next, we present a few assumptions for the controller $G_g(\cdot; A)$.

Denote the average matrix as $R := \frac{1}{N}\mathbb{1}\mathbb{1}^T$. Then we have the following assumptions on G_g:

A 1 (Control Signal Direction) *The output of G_g aligns with the direction that reduces the consensus error, that is:*

$$\langle (I - R) \cdot \mathbf{y}, G_g(\mathbf{y}; A) \rangle \geq C_g \cdot \|(I - R) \cdot \mathbf{y}\|^2, \ \forall\, \mathbf{y},$$

for some consent $C_g > 0$. Further, the global controller is an averaging algorithm, satisfying:

$$\langle \mathbb{1}, G_g(\mathbf{y}; A) \rangle = 0, \ \forall\, \mathbf{y}, \text{ or equivalently } \langle \mathbb{1}, u_g(t) \rangle = 0, \ \forall\, t.$$

A 2 (Linear Operator) *The controller G_g is a linear operator of $\mathbf{y}$, i.e.,*

$$a_1 \cdot G_g(\mathbf{y}; A) + a_2 \cdot G_g(\mathbf{y}'; A) = G_g(a_1 \cdot \mathbf{y} + a_2 \cdot \mathbf{y}'; A), \forall\, a_1, a_2 \in \mathbb{R}, \ \forall\, \mathbf{y}, \mathbf{y}' \in \mathbb{R}^{d_x+d_v}.$$

Let us comment on these assumptions. First, it is easy to check that A2 holds in most of the existing consensus algorithms. Second, A1 is also easy to satisfy. For example, when $G_g(\cdot; A)$ performs a weighted averaging, we have $G_g(\mathbf{y}; A) = (I - A^T W A) \cdot \mathbf{y}$ where W is a diagonal matrix containing the weights of the edges. It is easy to verify that, C_g is the second smallest eigenvalue of $I - A^T W A$, that is, $C_g = 1 - \lambda_2(A^T W A)$ where $\lambda_2(\cdot)$ denotes the second largest eigenvalue [4, 1].

By using A1, following the general analysis of averaging systems [23] we can prove that the GCFL behaves at least *as expected* – it will drive the agents to a consensus state. Note that the this result does not require the linearity in Assumption 2. This assumption is still important when we analyze the discretized system later.

2.3 The Local Computation Feedback Loop

The LCFL optimizes the local function $f_i(\cdot)$ for each agent. More specifically, for at time t, the ith local controller takes the local variables $x_i(t), v_i(t), z_i(t)$ as inputs and produces a local control signal. To describe the system, let us first denote the local controllers for different variables as:

$$\begin{aligned}u_{i,\ell,x}(t) &:= G_{\ell,x}(x_i(t), v_i(t), z_i(t); f_i), \ \forall\, i \in [N],\\ u_{i,\ell,v}(t) &:= G_{\ell,v}(x_i(t), v_i(t), z_i(t); f_i), \ \forall\, i \in [N],\\ u_{i,\ell,z}(t) &:= G_{\ell,z}(x_i(t), v_i(t), z_i(t); f_i), \ \forall\, i \in [N].\end{aligned}$$

Stacking them together, we can define the local controller $G_\ell(\cdot; f_i)$'s, as well as the output signals $u_{i,\ell}(t)$'s, as follows:

$$G_\ell(\cdot; f_i) := [G_{\ell,x}(\cdot; f_i); G_{\ell,x}(\cdot; f_i); G_{\ell,x}(\cdot; f_i)],\ \forall\, i \in [N]$$
$$u_{i,\ell}(t) := G_\ell(\cdot; f_i) = [u_{i,\ell,x}(t); u_{i,\ell,v}(t); u_{i,\ell,z}(t)],\ \forall\, i \in [N].$$

Further, we denote the concatenated local controller outputs as: $u_{\ell,x}(t) := [u_{1,\ell,x}(t); \ldots; u_{N,\ell,x}(t)]$, and define $u_{\ell,v}(t), u_{\ell,z}(t)$ similarly. Note that we have assumed that all the agents use the same local controller $G_\ell(\cdot;\cdot)$, but they are parameterized by different f_i's. After multiplied by the control gain $\eta_\ell(t) > 0$, the resulting signal will be combined with the output of GCFL, and be fed back to the local controllers.

Below, we present some general assumptions on the local computation controllers. For simplicity, we consider deterministic controllers.

A 3 (Lipschitz Smoothness) *The controller is Lipschitz smooth with constant L:*

$$\|G_\ell(x, v, z; f_i) - G_\ell(x', v', z'; f_i)\| \le L \,\|[x; v; z] - [x'; v'; z']\|\,,$$
$$\forall\, i \in [N], x, x' \in \mathbb{R}^{d_x}, v, v' \in \mathbb{R}^{d_v}, z, z' \in \mathbb{R}^{d_z}.$$

A 4 (Control Signal Direction and Size) *Assume that $x_i(t_0)$, $v_i(t_0)$ and $z_i(t_0)$ are initialized properly, and that the local controllers are designed such that the output of $G_{\ell,x}$ aligns with the gradient $\nabla f(\mathbf{x})$, that is, the following holds:*

$$\langle \nabla f_i(x_i(t)), u_{i,\ell,x}(t)\rangle \ge \alpha(t) \cdot \|\nabla f_i(x_i(t))\|^2\,,\ \forall\, t \ge t_0,$$

where $\alpha(t) > 0$ satisfies $\lim_{t\to\infty} \int_{\tau=t_0}^{t} \alpha(\tau)\mathrm{d}\tau \to \infty$. Further, for any given x_i, v_i, z_i, the size of the control signal is upped bounded by the size of the local gradient. That is, for some positive constants C_x, C_v and C_z, the following holds:

$$\|u_{i,\ell,x}\| \le C_x \cdot \|\nabla f_i(x_i)\|\,,\ \|u_{i,\ell,v}\| \le C_v \cdot \|\nabla f_i(x_i)\|\,,\ \|u_{i,\ell,z}\| \le C_z \cdot \|\nabla f_i(x_i)\|\,.$$

Let us comment on these assumptions. A3 assumes that the controller has Lipschitz smoothness, which is easy to verify for any given realizations of the local controllers. A4 abstracts the convergence property of the local optimizer. This assumption implies that the update direction $-u_{i,\ell,x}(t)$ points to a direction that decreases the local objective values. Note that in this assumption, we have assumed that x_i, v_i and z_i are initialized properly. This qualification is needed since, in some of the cases, improper initial values lead to non-convergence of the algorithm. For example, for accelerated gradient descent method [24, 13], $z_i(t_0)$ should be initialized as $\nabla f_i(x_i(t_0))$.

Using A4, we can also derive the convergence property of local controllers mainly follows that of the gradient flow algorithms; see, e.g., [25]. The result says that under A4, the LCFL behaves *as expected* – the agents can at least properly optimize their local problems. Note that the above result does not require the Lipschitz smoothness in A3. This assumption is still important when we analyze the discretized system later.

2.4 Convergence Properties

In this subsection, we analyze the convergence properties of the continuous-time feedback control system. Towards this end, let us define the energy function as follows:

$$\mathcal{E}(\mathbf{x}(t), \mathbf{v}(t), \mathbf{z}(t)) := f(\bar{\mathbf{x}}(t)) - f^\star + \frac{1}{2}\,\|(I - R)\cdot \mathbf{y}(t)\|^2\,,$$

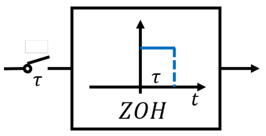
where $f^\star := \inf f(x)$ denotes the optimal objective value, $\bar{\mathbf{x}}(t) := \frac{1}{N}\mathbb{1}^T\mathbf{x}(t)$ denotes the average of $x_i(t)$'s. Then we have the following characterization for the dynamics of the energy function:

Theorem 1 *Assume that the derivative of the energy function $\mathcal{E}$ is upper bounded by:*

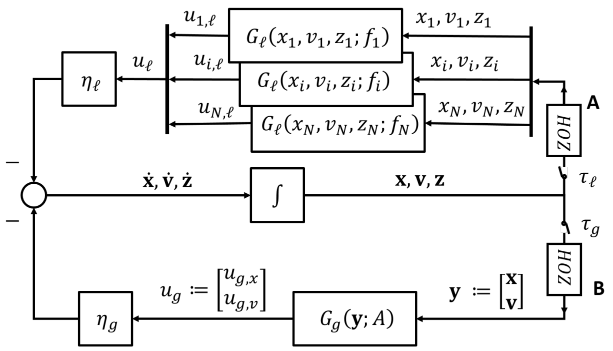
$$\dot{\mathcal{E}}(\mathbf{x}(t), \mathbf{v}(t), \mathbf{z}(t)) \le -\gamma_1(t) \cdot \left\| \frac{1}{N}\sum_{i=1}^{N} \nabla f_i(\bar{\mathbf{x}}(t))\right\|^2 - \gamma_2(t)\cdot \|(I-R)\cdot \mathbf{y}(t)\|^2\,, \tag{3}$$

where $\gamma_1(t), \gamma_2(t) > 0$. Then the continuous-time dynamic satisfies the following:

$$\min_t \left\| \frac{1}{N}\sum_{i=1}^{N} \nabla f_i(\bar{\mathbf{x}}(t))\right\|^2 + \min_t \|(I-R)\cdot \mathbf{y}(t)\|^2 \le \mathcal{O}\left(\max\left\{\frac{1}{\int_{t=0}^{T}\gamma_1(t)\mathrm{d}t}, \frac{1}{\int_{t=0}^{T}\gamma_2(t)\mathrm{d}t}\right\}\right).$$



(a) The discretization block that has a switch and a Zero-Order Hold.



$u_{1,\ell}$
$G_\ell(x_1, v_1, z_1; f_1)$
x_1, v_1, z_1
η_ℓ
u_ℓ
$u_{i,\ell}$
$G_\ell(x_i, v_i, z_i; f_i)$
x_i, v_i, z_i
$u_{N,\ell}$
$G_\ell(x_N, v_N, z_N; f_N)$
x_N, v_N, z_N
A
ZOH
τ_ℓ
$-$
$\dot{\mathbf{x}}, \dot{\mathbf{v}}, \dot{\mathbf{z}}$
$\int$
$\mathbf{x}, \mathbf{v}, \mathbf{z}$
$-$
τ_g
ZOH
B
$u_g := \begin{bmatrix} u_{g,x} \\ u_{g,v} \end{bmatrix}$
$\mathbf{y} := \begin{bmatrix} \mathbf{x} \\ \mathbf{v} \end{bmatrix}$
η_g
$G_g(\mathbf{y}; A)$

(b) Discretized system using ZOH on both control loops with two sampling time τ_g, τ_l

Figure 2: Discretization approach: a) Zero-Order Hold; b) discretized multi-agent control system.

The proof of this result is straightforward. By integrating $\dot{\mathcal{E}}(t)$ from $t = 0$ to T, we have

$$\int_{t=0}^{T} \gamma_2(t) \left\|(I - R) \cdot \mathbf{y}(t)\right\|^2 \mathrm{d}t + \int_{t=0}^{T} \gamma_1(t) \left\|\nabla f(\bar{\mathbf{x}}(t))\right\|^2 \leq \mathcal{E}(0) - \mathcal{E}(T),$$

divide both sides by $\int_{t=0}^{T} \gamma_1(t)\mathrm{d}t$ or $\int_{t=0}^{T} \gamma_1(t)\mathrm{d}t$, the prove is complete.

The above result provides us with a generic condition on the dynamics of the energy function, under which the convergence rate of the continuous-time system can be characterized. Despite being very simple and intuitive, this result is very useful for the following two main reasons. First, it suggests that the design of the (continuous-time) communication and computation controllers should follow the criteria set forth by condition (3), and, preferably, with large $\gamma_1(t)$ and $\gamma_2(t)$ so that the final convergence rate can be made faster; Second, if the continuous-time system satisfies (3), we will show that it is possible to discretize the system so that (3) can still be satisfied, but with different γ_1 and γ_2, and hence slightly different convergence characterization.

By now, we have built the double feedback-loop continuous-time control system, and provided a few basic assumptions that any reasonable (global consensus and local computation) feedback loop should satisfy. Further, we have identified a simple sufficient condition, under which it is relatively easy to analyze the convergence of the system (by mostly leveraging existing continuous-time analysis techniques). Now we are ready to dive into the main technical part of this work, which is to show that when the system is properly discretized, the resulting (discrete, and possibly mixed discrete and continuous) system can be used to model a few classes of decentralized learning algorithms, and they preserve the convergence behavior of their continuous-time counterpart.

3 System Discretization

In the previous section, we have mapped the decentralized optimization problem (1) to the double feedback-loop continuous-time system. It was shown that, we can identify a sufficient condition (3) to analyze decentralized algorithms for problem (1). Unfortunately, the analysis and results provided therein are only valid if the control signals are updated in continuous time, while in practice, optimization algorithms and communication protocols are implemented via discrete-time systems. Therefore, it is critical to understand, to what extent can the generic continuous-time analysis be extended to discrete time systems.

In this section, we discuss the impact of system discretization on the proposed double-feedback control system. Since our continuous-time control system involves *two* different feedback loops GCFL and LCFL, special attention will be given to the two loops that are discretized differently. Interestingly, we will show that many state-of-the-art algorithms for decentralized learning can be precisely mapped to some versions of discretized double-feedback control system, by properly choosing a specific discretization scheme, and by specializing the global and local controllers.

3.1 Modeling the Discretization

Typically, a continuous-time system is discretized by using a switch that samples the input with sample time τ, followed by a Zero-Order Hold (ZOH) that keeps the sampled signal as constant between the consecutive sampling instances [26]; see Figure 2a for an illustration. Note that the continuous-time system is equivalent to the case where the sampling time $\tau = 0$ and the switch is constantly on.

Let us begin by using such a block to discretize the proposed double-feedback continuous-time control system. By examining Fig. 2b, we see that the discretization can happen at the two points A and B, where the local states are about to enter the controllers. It is important to observe that, depending on which place (or places) that the discretization blocks are implemented, and depending on the actual sampling rate for each of the discretization block, the original continuous system can be discretized in many different ways. In a high-level, each of these discretization scheme will corresponds to a *multi-rate* control system, in which different parts of the system runs on different sampling rates. To precisely understand the effect of such kind of multi-rate system, let us define the *sampling intervals* for the GCFL and LCFL as τ_g and τ_ℓ, respectively.

3.2 Decentralized Algorithms as Multi-Rate Discretized Systems

In this section, we make the connection between different *classes* of decentralized algorithms and different discretization patterns. For convenience, let t_k denote the times at which the inputs of the ZOHs get sampled by *both* the global and local controllers. Note that when the sampling interval is zero, the corresponding ZOH samples all the time.

Case 1 ($\tau_g > 0, \tau_\ell = 0$): In this case, the local updates are continuous and the global consensus signal is updated every τ_g time interval. The system becomes

$$\begin{aligned}\dot{\mathbf{v}}(t) &= -\eta_g(t)\cdot u_{g,v}(t_k) - \eta_\ell(t)\cdot u_{\ell,v}(t)\\ \dot{\mathbf{x}}(t) &= -\eta_g(t)\cdot u_{g,x}(t_k) - \eta_\ell(t)\cdot u_{\ell,x}(t)\\ \dot{\mathbf{z}}(t) &= -\eta_\ell(t)\cdot u_{\ell,z}(t).\end{aligned} \tag{4}$$

By A4, we know that with only u_ℓ, the dynamic system finds a stationary point of the local problem that $\|\nabla f_i(x_i)\|^2 = 0$. So with (4), the dynamic system finds a stationary point that $u_\ell + \frac{\eta_g(t)}{\eta_\ell(t)} u_g = 0$, which is the stationary solution of the following problem for each agent:

$$f_i'(x_i(t)) := f_i(x_i(t)) + \frac{\eta_g(t)}{\eta_l(t)} \langle u_{i,g}(t_k), y_i(t)\rangle .$$

Therefore, from t_0 to $t_0 + \tau_g$, each agent minimizes a perturbed local problem to $\gamma(\tau)$ accuracy. This system has the same form as the distributed algorithms that require to solve some local problems to a given accuracy, before any local communication steps take place; see for examples FedProx [9], FedPD [11] and NEXT [7].

Case 2 ($\tau_g = 0, \tau_\ell > 0$): In this case, the global consensus signal is continuous and the local updates are updated every τ_ℓ time interval. The system becomes

$$\begin{aligned}\dot{\mathbf{v}}(t) &= -\eta_g(t)\cdot u_{g,v}(t) - \eta_\ell(t)\cdot u_{\ell,v}(t_k)\\ \dot{\mathbf{x}}(t) &= -\eta_g(t)\cdot u_{g,x}(t) - \eta_\ell(t)\cdot u_{\ell,x}(t_k)\\ \dot{\mathbf{z}}(t) &= -\eta_\ell(t)\cdot u_{\ell,z}(t_k).\end{aligned} \tag{5}$$

Similar to Case I, between τ_ℓ time interval, the dynamic system finds the first-order stationary point the following problem

$$\left\|(I-R)\mathbf{y} + \frac{\eta_\ell(t)}{\eta_g(t)} u_{\ell,y}(t_k)\right\|^2$$

to certain accuracy. This system has the same form as the algorithms that require to solve some networked problems to a certain accuracy, see for example [27, 28, 12].

Case 3 ($\tau_g = \tau_\ell > 0$): In this case, the system is discretized with the same sampling time. The update of the system can be written as:

$$\begin{aligned}\mathbf{x}(t_{k+1}) &= \mathbf{x}(t_k) - \eta_\ell'(t_k)\cdot u_{\ell,x}(t_k) - \eta_g'(t_k)\cdot u_{g,x}(t_k),\\ \mathbf{v}(t_{k+1}) &= \mathbf{v}(t_k) - \eta_\ell'(t_k)\cdot u_{\ell,v}(t_k) - \eta_g'(t_k)\cdot u_{g,v}(t_k),\\ \mathbf{z}(t_{k+1}) &= \mathbf{z}(t_k) - \eta_g'(t_k)\cdot u_{\ell,z}(t_k),\end{aligned} \tag{6}$$

where $\eta_\ell'(t_k) = \int_{t_k}^{t_k+\tau_g} \eta_\ell(t)\mathrm{d}t$, $\eta_g'(t_k) = \int_{t_k}^{t_k+\tau_g} \eta_g(t)\mathrm{d}t$. The above updates can be shown to be equivalent to many existing DO algorithms, such as DGD, D^2 [29], DLM, which perform one step local update, followed by one step of communication.

Case	$\tau_\ell, \tau_\mathbf{g}$	Communication	Computation	Related Algorithm
I	$\tau_g > 0, \tau_l = 0$	Slow	Continuous	NEXT [7], FedProx [9]
II	$\tau_g = 0, \tau_l > 0$	Continuous	Slow	GPDA [28]
III	$\tau_g = \tau_l > 0$	Same rate		DGD [4], DGT [5]
IV	$\tau_g > \tau_l > 0$	Slow	Fast	Scaffold [10], NIDS [30]
V	$\tau_l > \tau_g > 0$	Fast	Slow	AGD [13], xFilter [12]

Table 1: Summary of different discretization settings, and their corresponding distributed learning algorithms.

Case 4 ($\tau_g > \tau_l > 0$): In this case, we assume that $\tau_g = Q \cdot \tau_l$, which means that each agent performs Q times local computation steps between two communication steps. This update strategy has the same spirit as the class of (horizontal) FL algorithms [8].

Case 5 ($\tau_l > \tau_g > 0$): In this case, we assume that $\tau_l = K \cdot \tau_g$, which means that the agents perform K times communication steps before two local computation steps. This update strategy is similar to the line of works that are trying to achieve optimal communication complexities [12, 30].

We summarize the above discussion in Table 1 and provide some examples of the algorithms. Note that the above discussion about relations of algorithms and discretization settings is still a bit vague, but later in Appendix A and B, we will provide specific examples to showcase how one can precisely map an existing algorithm into a discretization setting.

Such kinds of connections are useful in the following sense: First, it suggests that *different* classes of algorithms may be rooted in *the same* continuous-time system, so they are likely to share some common properties, and it is plausible that they can be covered by a single analysis framework. Second, by properly utilizing such kinds of connections, we can develop a systematic way of designing new, and application-specific algorithms. More specifically, we can begin by designing and analyzing a continuous-time control system (say, with a specific controller), then choose a desirable discretization scheme, and transfer the theoretical results to such a particular setting. Therefore, it will be important to have a systematic way of transferring the theoretical results from the continuous-time system to different discretization settings.

3.3 Convergence Result of Discretized Multi-Rate Systems

For a given distributed algorithm, we can first find its continuous-time counterpart, then perform discretization based on the requirements of each part of the system. This procedure allows the new algorithm to share some desirable properties of the original algorithm. However, the discretization procedure will introduce instability to the system as the sampled control signal will deviate from the continuous-time control signal. As the sampling interval increases, the deviation also increases. Understanding how the deviations introduced by different discretization schemes affect the system is crucial to transferring the theoretical results from the continuous-time system to discretized systems. The following result provides a way to analyze the convergence for all different discretization schemes.

Theorem 2 (Dynamics of $\mathcal{E}$ for discretized system) *Under assumptions A1-A4 and the choice of η_ℓ and η_g in Lemma 1, and consider the discretize-time system with $\tau_\ell \geq 0, \tau_g \geq 0$. Then we have the following:*

$$\dot{\mathcal{E}}(t) \leq -\left(\frac{\gamma_1(t)}{2} - C_3(t)\right) \left\|\sum_{i=1}^{N} \nabla f_i(\bar{\mathbf{x}}(t))\right\|^2 - \left(\frac{\gamma_2(t)}{2} - C_4(t)\right) \|(I-R)\mathbf{y}(t)\|^2, \tag{7}$$

where $C_3(t)$ and $C_4(t)$ are some constants depending on $N, C_g, L, \eta_\ell(t), \eta_g(t), \tau_\ell, \tau_g, K, Q$.

This result indicates that by proper choice of τ_ℓ, τ_g, K, Q, such that $\left(\frac{\gamma_1(t)}{2} - C_3(t)\right) > 0$ and $\left(\frac{\gamma_2(t)}{2} - C_4(t)\right) > 0$, the discretized algorithm preserves the convergence rate of the continuous-time system and only slows down by a constant factor $\max\left\{\gamma_1(t) \Big/ \left(\frac{\gamma_1(t)}{2} - C_3(t)\right), \gamma_2(t) \Big/ \left(\frac{\gamma_2(t)}{2} - C_4(t)\right)\right\}$. The detailed convergence analyses and discussions are omitted due to page limitation.

References

[1] T.-H. Chang, M. Hong, H.-T. Wai, X. Zhang, and S. Lu, "Distributed learning in the nonconvex world: From batch data to streaming and beyond," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 26–38, 2020.

[2] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[3] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, 2009.

[4] K. Yuan, Q. Ling, and W. Yin, "On the convergence of decentralized gradient descent," *SIAM Journal on Optimization*, vol. 26, no. 3, pp. 1835–1854, 2016.

[5] K. Yuan, W. Xu, and Q. Ling, "Can primal methods outperform primal-dual methods in decentralized dynamic optimization?" *arXiv preprint arXiv:2003.00816*, 2020.

[6] Q. Ling, W. Shi, G. Wu, and A. Ribeiro, "Dlm: Decentralized linearized alternating direction method of multipliers," *IEEE Transactions on Signal Processing*, vol. 63, no. 15, pp. 4051–4064, 2015.

[7] P. Di Lorenzo and G. Scutari, "Next: In-network nonconvex optimization," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 2, no. 2, pp. 120–136, 2016.

[8] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečnỳ, S. Mazzocchi, H. B. McMahan *et al.*, "Towards federated learning at scale: System design," *arXiv preprint arXiv:1902.01046*, 2019.

[9] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *arXiv preprint arXiv:1812.06127*, 2018.

[10] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "Scaffold: Stochastic controlled averaging for federated learning," in *International Conference on Machine Learning*. PMLR, 2020, pp. 5132–5143.

[11] X. Zhang, M. Hong, S. Dhople, W. Yin, and Y. Liu, "Fedpd: A federated learning framework with optimal rates and adaptivity to non-iid data," *arXiv preprint arXiv:2005.11418*, 2020.

[12] H. Sun and M. Hong, "Distributed non-convex first-order optimization and information processing: Lower complexity bounds and rate optimal algorithms," *IEEE Transactions on Signal processing*, vol. 67, no. 22, pp. 5912–5928, 2019.

[13] H. Ye, L. Luo, Z. Zhou, and T. Zhang, "Multi-consensus decentralized accelerated gradient descent," *arXiv preprint arXiv:2005.00797*, 2020.

[14] R. Rossi and G. Savaré, "Gradient flows of non convex functionals in hilbert spaces and applications," *ESAIM: Control, Optimisation and Calculus of Variations*, vol. 12, no. 3, pp. 564–614, 2006.

[15] J. Wang and N. Elia, "A control perspective for centralized and distributed convex optimization," in *2011 50th IEEE conference on decision and control and European control conference*. IEEE, 2011, pp. 3800–3805.

[16] A. Sundararajan, *Analysis and Design of Distributed Optimization Algorithms*. The University of Wisconsin-Madison, 2021.

[17] L. Lessard, B. Recht, and A. Packard, "Analysis and design of optimization algorithms via integral quadratic constraints," *SIAM Journal on Optimization*, vol. 26, no. 1, pp. 57–95, 2016.

[18] B. Hu and L. Lessard, "Control interpretations for first-order optimization methods," in *2017 American Control Conference (ACC)*. IEEE, 2017, pp. 3114–3119.

[19] M. Muehlebach and M. Jordan, "A dynamical systems perspective on nesterov acceleration," in *International Conference on Machine Learning*, 2019, pp. 4656–4662.

[20] B. Swenson, R. Murray, H. V. Poor, and S. Kar, "Distributed gradient flow: Nonsmoothness, nonconvexity, and saddle point evasion," *IEEE Transactions on Automatic Control*, 2021.

[21] G. França, D. P. Robinson, and R. Vidal, "A dynamical systems perspective on nonsmooth constrained optimization," *arXiv preprint arXiv:1808.04048*, 2018.

[22] B. Swenson, R. Murray, H. V. Poor, and S. Kar, "Distributed gradient descent: Nonconvergence to saddle points and the stable-manifold theorem," in *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2019, pp. 595–601.

[23] A. Olshevsky and J. N. Tsitsiklis, "Convergence speed in distributed consensus and averaging," *SIAM journal on control and optimization*, vol. 48, no. 1, pp. 33–55, 2009.

[24] S. Bubeck, Y. T. Lee, and M. Singh, "A geometric alternative to nesterov's accelerated gradient descent," *arXiv preprint arXiv:1506.08187*, 2015.

[25] A. Orvieto and A. Lucchi, "Continuous-time models for stochastic optimization algorithms," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[26] B. C. Kuo, "Digital control systems," 1980.

[27] M. Hong, D. Hajinezhad, and M.-M. Zhao, "Prox-pda: The proximal primal-dual algorithm for fast distributed nonconvex optimization and learning over networks," in *International Conference on Machine Learning*, 2017, pp. 1529–1538.

[28] M. Hong, M. Razaviyayn, and J. Lee, "Gradient primal-dual algorithm converges to second-order stationary solution for nonconvex distributed optimization over networks," in *International Conference on Machine Learning*. PMLR, 2018, pp. 2009–2018.

[29] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 5336–5346.

[30] Z. Li, W. Shi, and M. Yan, "A decentralized proximal-gradient method with network independent step-sizes and separated convergence rates," *IEEE Transactions on Signal Processing*, vol. 67, no. 17, pp. 4494–4506, 2019.

[31] S. Lu, X. Zhang, H. Sun, and M. Hong, "Gnsd: A gradient-tracking based nonconvex stochastic algorithm for decentralized optimization," in *2019 IEEE Data Science Workshop (DSW)*. IEEE, 2019, pp. 315–321.

[32] H. Sun, S. Lu, and M. Hong, "Improving the sample and communication complexity for decentralized non-convex optimization: Joint gradient estimation and tracking," in *International Conference on Machine Learning*. PMLR, 2020, pp. 9217–9228.