

# ProxSkip: Yes! Local Gradient Steps Provably Lead to Communication Acceleration! Finally!

arXiv:2202.09357v1 [cs.LG] 18 Feb 2022

**ProxSkip: Yes! Local Gradient Steps Provably Lead to Communication Acceleration! Finally!**

Konstantin Mishchenko<sup>1</sup> Grigory Malinovsky<sup>2</sup> Sebastian Stich<sup>3</sup> Peter Richtárik<sup>2</sup>

**Abstract**

We introduce ProxSkip, a surprisingly simple and provably efficient method for minimizing the sum of a smooth ( $f$ ) and an expensive non-smooth proximable ( $\psi$ ) function. The canonical approach to solving such problems is proximal gradient descent (ProxGD), which is based on the evaluation of the gradient of  $f$  and the prox operator of  $\psi$ . ProxSkip is specifically interested in the regime in which the evaluation of  $f$  is costly relative to the evaluation of  $\psi$ , and it allows skipping the gradient evaluations. ProxSkip allows for the expensive prox operator to be skipped in many iterations: while it is not possible to prove a linear rate of convergence in general, we show that the number of prox evaluations is  $O(\sqrt{\log(1/\epsilon)})$  only. Our main motivation is to reduce communication costs in distributed optimization of the gradient-type algorithms. In this context, ProxSkip offers an effective alternative to the state-of-the-art methods. Unlike other local gradient-type methods, such as FedAvg, SCAFFOLD, S-local-GD and FedUp, which are either significantly slower or worse than, or at best matching, that of vanilla GD in the heterogeneous data regime, we obtain a provable and large improvement without any heterogeneity-bounding assumptions.

**1. Introduction**

We study optimization problems of the form

$$\min_x (f(x) + \psi(x)). \quad (1)$$

where  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  is a smooth function, and  $\psi : \mathbb{R}^d \rightarrow \mathbb{R}$  is a closed and convex function.

Such problems are ubiquitous and appear in numerous applications associated with virtually all areas of science and engineering, including signal processing (Combettes & Pesquet, 2011), image processing (Lake, 2020), data science (Furukawa & Saito, 2014) and machine learning (Dhillon, Shwartz & Ben-David, 2014).

**1.1. Proximal gradient descent**

One of the most canonical methods for solving (1), often used as the basis for further extensions and improvements, is proximal gradient descent (ProxGD), also known as the forward-backward splitting algorithm (Bauschke & Nesterov, 2004). This method solves (1) via the iterative process defined by

$$x_{t+1} = \text{prox}_{\psi}(x_t - \eta_t \nabla f(x_t)), \quad (2)$$

where  $\eta_t > 0$  is a suitable stepsize at time  $t$ , and  $\text{prox}_{\psi}(\cdot) : \mathbb{R}^d \rightarrow \mathbb{R}^d$  is the proximity operator of  $\psi$ , defined via

$$\text{prox}_{\psi}(x) = \arg \min_y \left[ \frac{1}{2} \|y - x\|^2 + \psi(y) \right]. \quad (3)$$

It is typically assumed that the proximity operator (3) can be evaluated in closed form, which means that the iteration (2) defining ProxGD can be performed exactly. ProxGD is most suited to problems where the prox operator is relatively cheap to evaluate, so the bottleneck of (2) is in the forward step (i.e., computation of the gradient  $\nabla f$ ) rather than in the backward step (i.e., evaluation of  $\text{prox}_{\psi}$ ). This is true for a wide range of operators, including the  $L_1$  norm ( $\psi(x) = \|x\|_1$ ), the  $L_2$  norm ( $\psi(x) = \|x\|_2$ ), and classical quadratic regularizers. For more examples, we refer the reader to the books (Punith & Boyd, 2014; Beck, 2017).

**1.2. Expensive proximability operators**

However, in this work we are interested in the situation when the evaluation of the proximity operator is expensive. That is, we assume that the computation of  $\text{prox}_{\psi}(\cdot)$  is prohibitively expensive relative to the evaluation of the gradient of  $f$  (the forward step).

A conceptually simple yet rich class of expensive proximity operators arises from regularizers  $\psi$  encoding a

<sup>1</sup> Please accept our apologies, our excitement apparently spilled over into the title. If we were to write a more scholarly title for this work, it would be "Breaking the Barrier of Local Gradient Methods".

<sup>2</sup> CNRS, INRIA, Institut Élie Cartan de Lorraine, Computer Science, King Abdullah University of Science and Technology, Thawali, Saudi Arabia. CISPA Helmholtz Center for Information Security, Saarbrücken, Germany. Correspondence to: Peter Richtárik (<[peter.richtarik@kaust.edu.sa](mailto:peter.richtarik@kaust.edu.sa)>).

<sup>3</sup> CISPA, INRIA, Institut Élie Cartan de Lorraine, Computer Science, King Abdullah University of Science and Technology, Thawali, Saudi Arabia. CISPA Helmholtz Center for Information Security, Saarbrücken, Germany. Correspondence to: Peter Richtárik (<[peter.richtarik@kaust.edu.sa](mailto:peter.richtarik@kaust.edu.sa)>).

Peter Richtárik  
Baidu @ April 26, 2022

Konstantin Mishchenko, Grigory Malinovsky, Sebastian Stich, Peter Richtárik  
**ProxSkip: Yes! Local Gradient Steps Provably Lead to Communication Acceleration! Finally!**  
arXiv:2202.09357, 2022



# ProxSkip: Yes! Local Gradient Steps Provably Lead to Communication Acceleration! Finally!<sup>†</sup>

Konstantin Mishchenko<sup>1</sup> Grigory Malinovsky<sup>2</sup> Sebastian Stich<sup>3</sup> Peter Richtárik<sup>2</sup>

## Abstract

We introduce **ProxSkip**—a surprisingly simple and provably efficient method for minimizing the sum of a smooth ( $f$ ) and an expensive nonsmooth proximable ( $\psi$ ) function. The canonical approach to solving such problems is via the proximal gradient descent (**ProxGD**) algorithm, which is based on the evaluation of the gradient of  $f$  and the prox operator of  $\psi$  in each iteration. In this work we are specifically interested in the regime in which the evaluation of prox is costly relative to the evaluation of the gradient, which is the case in many applications. **ProxSkip** allows for the expensive prox operator to be skipped in most iterations, so its iteration complexity is  $\mathcal{O}(\kappa \log^{1/\epsilon})$ , where  $\kappa$  is the condition number of  $f$ , the number of evaluations is  $\mathcal{O}(\sqrt{\kappa} \log^{1/\epsilon})$  only. Our motivation comes from federated learning, where the evaluation of the gradient operator corresponds to a local **GD** step independently on all workers, and evaluation of prox corresponds to (expensive) communication in the form of gradient exchange. In this context, **ProxSkip** offers a provable acceleration of communication costs. Unlike other local gradient-type methods such as **FedAvg**, **SCAFFOLD**, **S-Local-GD** and others, whose theoretical communication complexity is worse than, or at best matching, that of **GD** in the heterogeneous data regime, we offer a provable and large improvement with heterogeneity-bounding assumptions.

where  $f: \mathbb{R}^d \rightarrow \mathbb{R}$  is a smooth function, and  $\psi: \mathbb{R}^d \rightarrow \mathbb{R} \cup \{+\infty\}$  is a proper, closed and convex regularizer.

Such problems are ubiquitous, and appear in numerous applications associated with virtually all areas of science and engineering, including signal processing (Combettes & Pesquet, 2009), image processing (Luke, 2020), data science (Parikh & Boyd, 2014) and machine learning (Shalev-Shwartz & Ben-David, 2014).

### 1.1. Proximal gradient descent

<sup>†</sup> Please accept our apologies, our excitement apparently spilled over into the title. If we were to choose a more scholarly title for this work, it would be *ProxSkip: Breaking the Communication Barrier of Local Gradient Methods.*

## 1. Introduction

We study optimization problems of the form

$$\min_{x \in \mathbb{R}^d} f(x) + \psi(x), \quad (1)$$

$\text{prox}_{\gamma\psi}$ . This is the case for many regularizers, including the  $L_1$  norm ( $\psi(x) = \|x\|_1$ ), the  $L_2$  norm ( $\psi(x) = \|x\|_2^2$ ), and elastic net (Zhou & Hastie, 2005). For many further examples, we refer the reader to the books (Parikh & Boyd, 2014; Beck, 2017).

## 1.2. Expensive proximity operators

However, in this work we are interested in the situation when the evaluation of the *proximity operator is expensive*. That is, we assume that the computation of  $\text{prox}_{\gamma\psi}$  (the backward step) is costly relative to the evaluation of the gradient of  $f$  (the forward step).

A conceptually simple yet rich class of expensive proximity operators arises from regularizers  $\psi$  encoding a

<sup>†</sup> Please accept our apologies, our excitement apparently spilled over into the title. If we were to choose a more scholarly title for this work, it would be *ProxSkip: Breaking the Communication Barrier of Local Gradient Methods.*

# Alternative Title

# Coauthors



**Konstantin Mishchenko**



**Grigory Malinovsky**

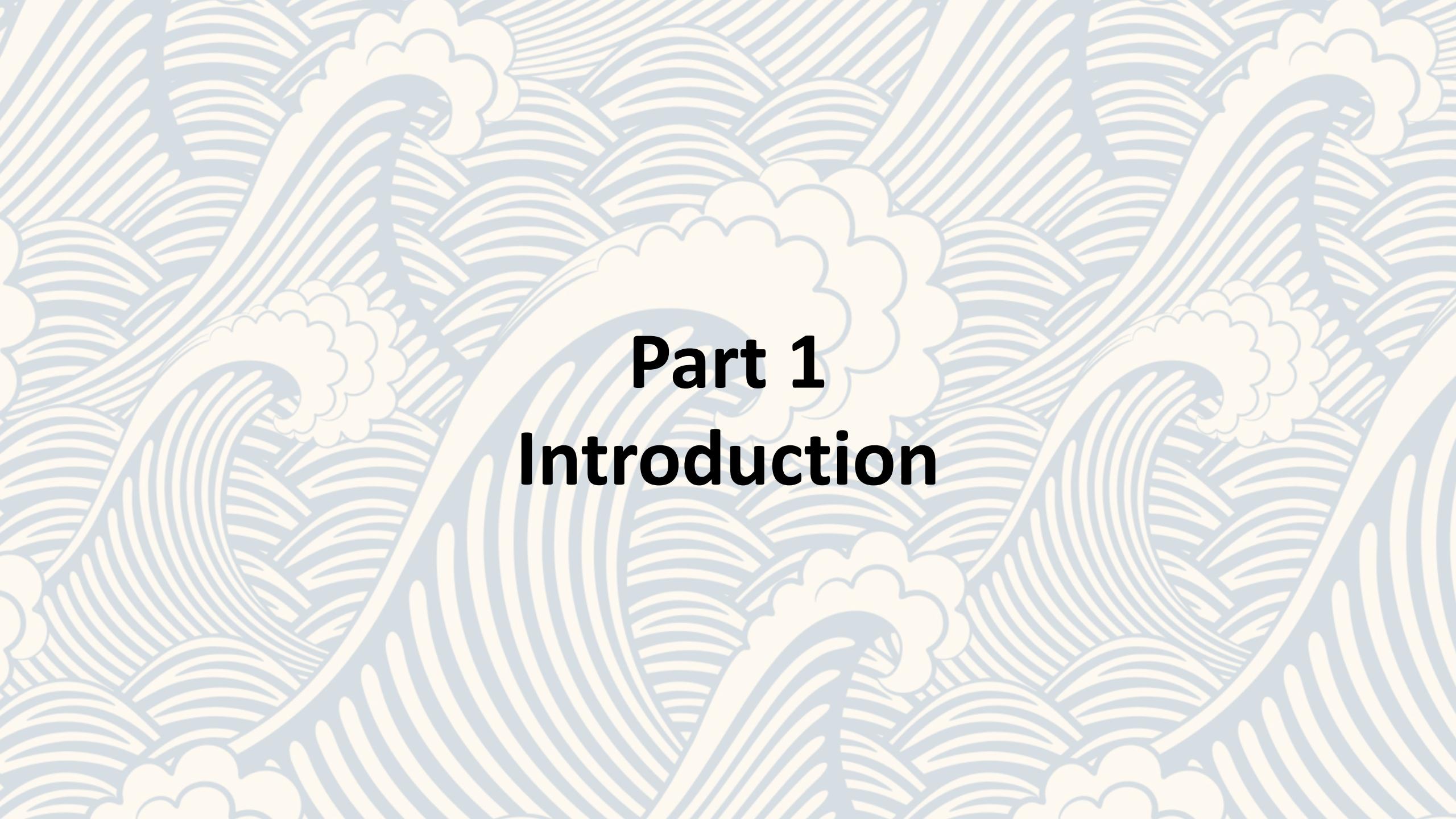


**Sebastian Stich**



# Outline of the Talk

1. Introduction
2. Consensus Reformulation
3. Proximal Gradient Descent
- 4. ProxSkip: Algorithm**
- 5. ProxSkip: Theory**
6. Experiments
7. Extensions



# **Part 1**

# **Introduction**

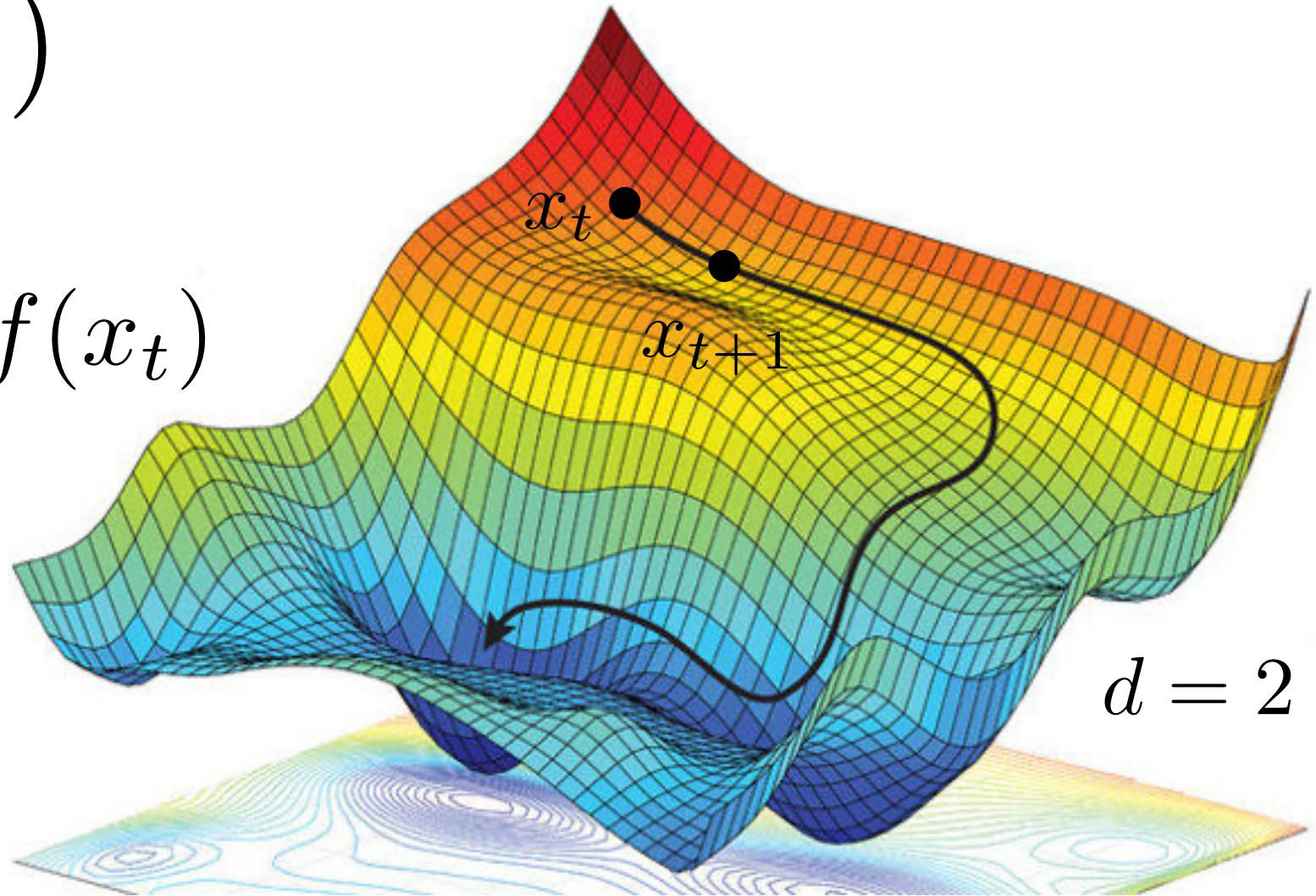
# **Gradient Descent**

# Gradient Descent

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

$$x_{t+1} = x_t - \gamma \nabla f(x_t)$$

Stepsize / Learning rate



# Distributed Gradient Descent

# Federated Training of a Supervised Machine Learning Model

$$\min_{x \in \mathbb{R}^d} f(x) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n f_i(x)$$

# model parameters / features

Loss on local data  $\mathcal{D}_i$  stored on device  $i$

$$f_i(x) = \mathbb{E}_{\xi \sim \mathcal{D}_i} f_{i,\xi}(x)$$

The datasets  $\mathcal{D}_1, \dots, \mathcal{D}_n$  can be arbitrarily heterogeneous

# devices / machines

Optimization problem:

$$\min_{x \in \mathbb{R}^d} f(x) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n f_i(x)$$

# Distributed Gradient Descent

$$\begin{aligned} x_{t+1} &= x_t - \gamma \nabla f(x_t) \\ &= x_t - \gamma \frac{1}{n} \sum_{i=1}^n \nabla f_i(x_t) \end{aligned}$$

$f(x) = \frac{1}{n} \sum_{i=1}^n f_i(x_t)$

$d$ -dimensional gradient computed by machine  $i$

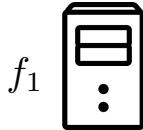
Optimization problem:

$$\min_{x \in \mathbb{R}^d} f(x) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n f_i(x)$$

# Distributed Gradient Descent

(Each worker performs 1 GD step using its local function, and the results are averaged)

Worker 1

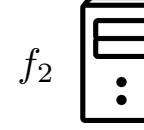


Receive  $x_t$  from the server

$$x_{1,t} = x_t$$

$$x_{1,t+1} = x_{1,t} - \gamma \nabla f_1(x_{1,t})$$

Worker 2

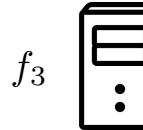


Receive  $x_t$  from the server

$$x_{2,t} = x_t$$

$$x_{2,t+1} = x_{2,t} - \gamma \nabla f_2(x_{2,t})$$

Worker 3



Receive  $x_t$  from the server

$$x_{3,t} = x_t$$

$$x_{3,t+1} = x_{3,t} - \gamma \nabla f_3(x_{3,t})$$

Server



$$x_{t+1} = \frac{1}{3} \sum_{i=1}^3 x_{i,t+1}$$

Broadcast  $x_{t+1}$  to the workers

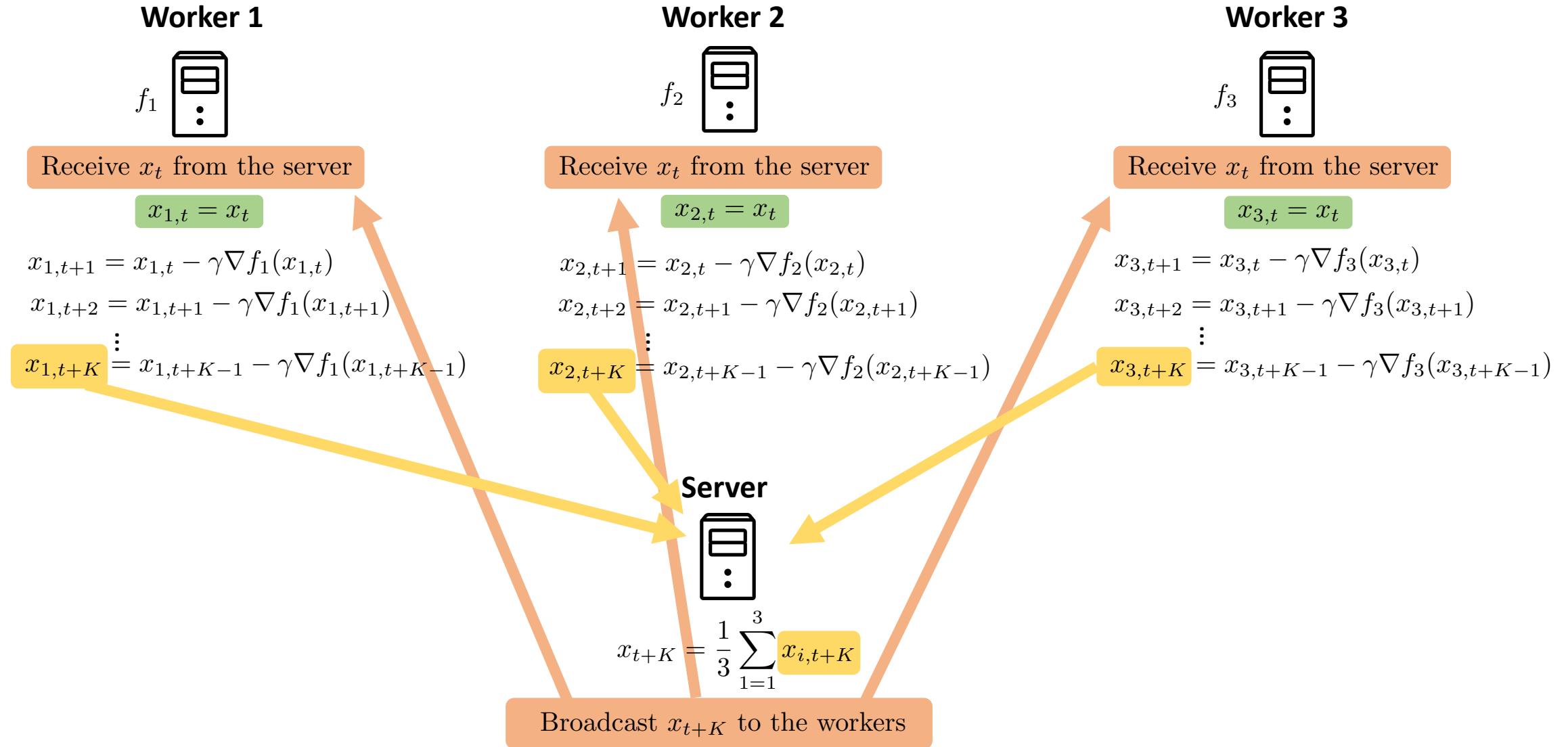
# Distributed Local Gradient Descent

# Distributed Local Gradient Descent

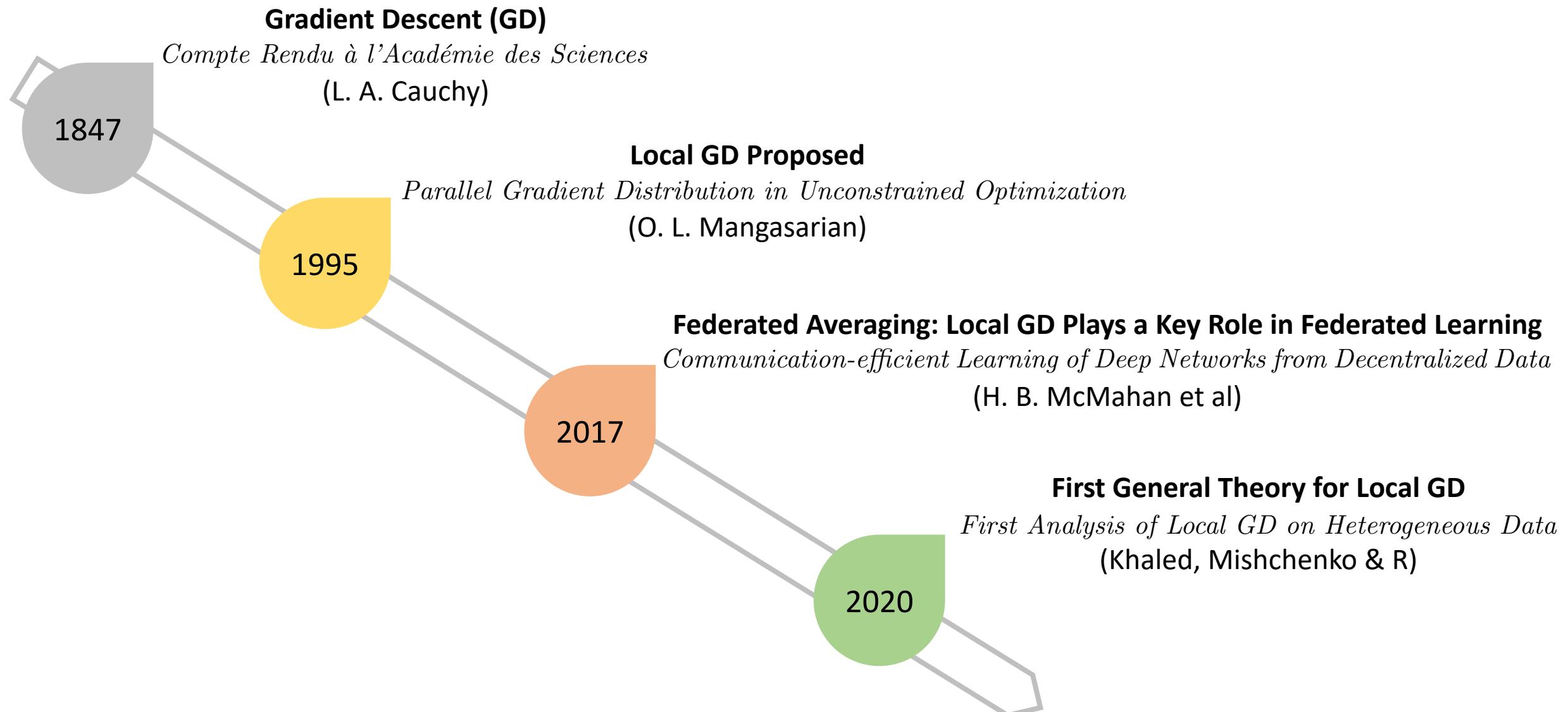
(Each worker performs  $K$  GD steps using its local function, and the results are averaged)

Optimization problem:

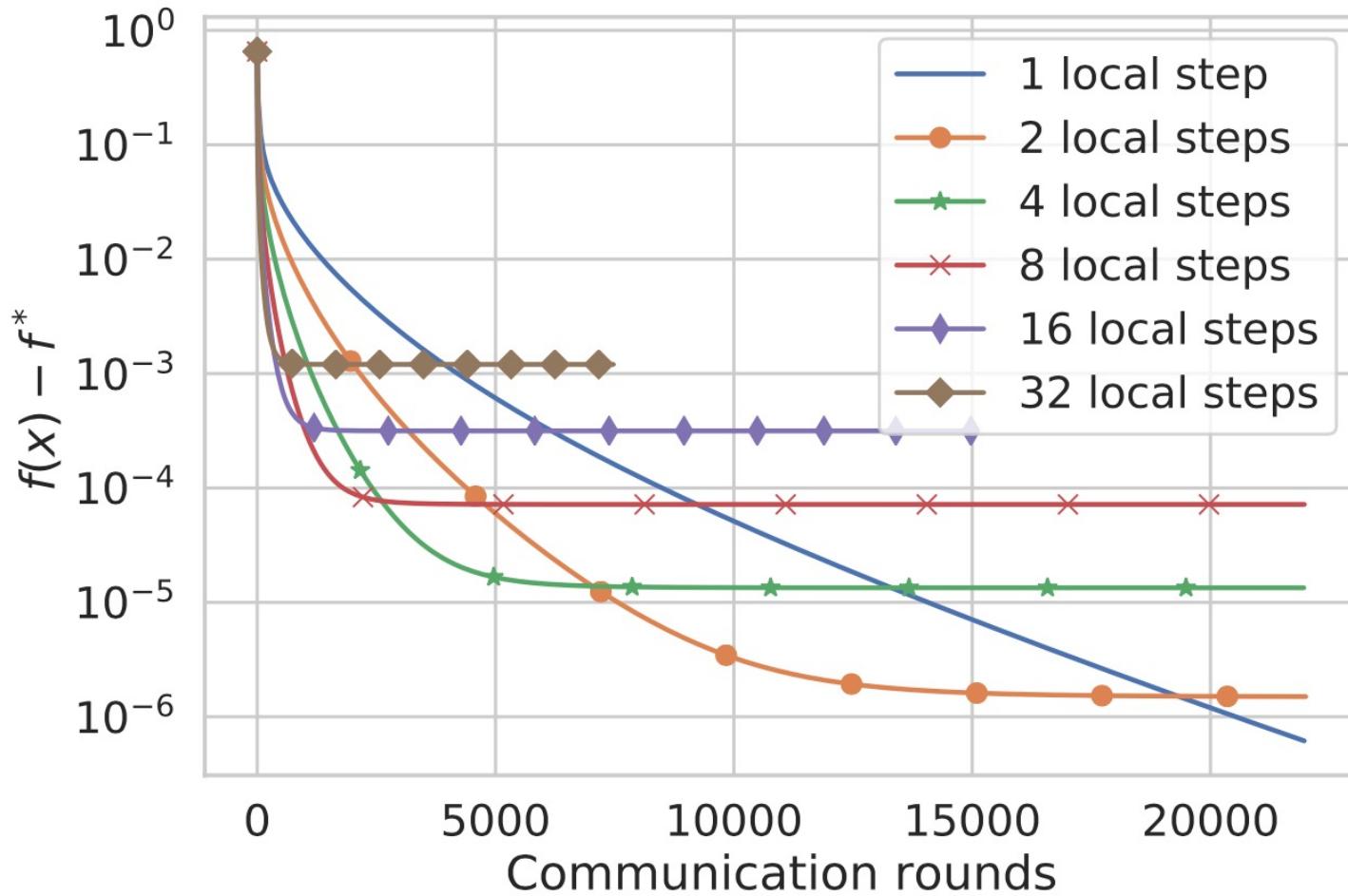
$$\min_{x \in \mathbb{R}^d} f(x) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n f_i(x)$$



# From GD to Local GD



# What do the Local Steps do?



Plot taken from:



Ahmed Khaled, Konstantin Mishchenko, Peter Richtárik

First Analysis of Local GD on Heterogeneous Data

NeurIPS 2019 Workshop on Federated Learning for Data Privacy and Confidentiality, 2019

L2-regularized logistic regression  
LibSVM mushrooms dataset

# **Linearly Converging Local GD Methods**

# Local GD with GD-like (=Linear) Convergence

2020

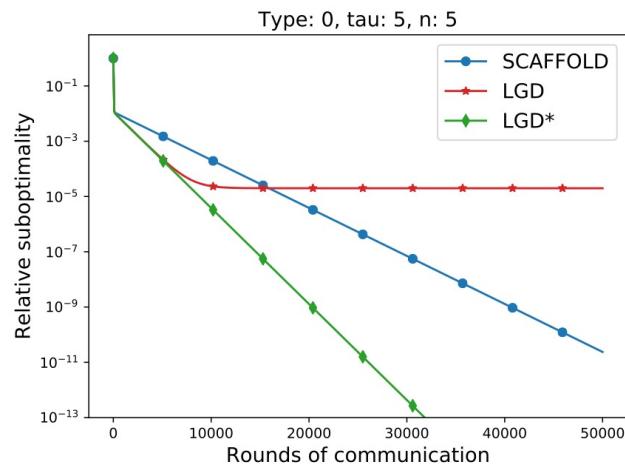
## SCAFFOLD

*Scaffold: Stochastic Controlled Averaging for Federated Learning*  
(Karimireddy, Kale, Mohri, Reddi, Stich, Suresh)

2021

## S-Local-GD

*Local SGD: Unified Theory and New Efficient Methods*  
(Gorbunov, Hanzely & R)



2021

## FedLin

*Linear Convergence in Federated Learning...*  
(Mitra, Jaafar, Pappas, Hassani)

# Key Theoretical Problem in Federated Learning

Local GD and its variants are of key importance in federated learning. They work better than GD in practice in terms of communication efficiency.

Why?

Do Multiple Local GD Steps Provably Help?  
No Current Theoretical Result Supports This.

# Federated Learning: ProxSkip vs Baselines

Table 1. The performance of federated learning methods employing multiple local gradient steps in the strongly convex regime.

method	# local steps per round	# floats sent per round	stepsize on client $i$	linear rate?	# rounds	rate better than GD?
GD (Nesterov, 2004)	1	$d$	$\frac{1}{L}$	✓	$\tilde{\mathcal{O}}(\kappa)$ <sup>(c)</sup>	✗
LocalGD (Khaled et al., 2019; 2020)	$\tau$	$d$	$\frac{1}{\tau L}$	✗	$\mathcal{O}\left(\frac{G^2}{\mu n \tau \varepsilon}\right)$ <sup>(d)</sup>	✗
Scaffold (Karimireddy et al., 2020)	$\tau$	$2d$	$\frac{1}{\tau L}$ <sup>(e)</sup>	✓	$\tilde{\mathcal{O}}(\kappa)$ <sup>(c)</sup>	✗
S-Local-GD <sup>(a)</sup> (Gorbunov et al., 2021)	$\tau$	$d < \# < 2d$ <sup>(f)</sup>	$\frac{1}{\tau L}$	✓	$\tilde{\mathcal{O}}(\kappa)$	✗
FedLin <sup>(b)</sup> (Mitra et al., 2021)	$\tau_i$	$2d$	$\frac{1}{\tau_i L}$	✓	$\tilde{\mathcal{O}}(\kappa)$ <sup>(c)</sup>	✗
Scaffnew <sup>(g)</sup> (this work) for any $p \in (0, 1]$	$\frac{1}{p}$ <sup>(h)</sup>	$d$	$\frac{1}{L}$	✓	$\tilde{\mathcal{O}}\left(p\kappa + \frac{1}{p}\right)$ <sup>(c)</sup>	✓ (for $p > \frac{1}{\kappa}$ )
Scaffnew <sup>(g)</sup> (this work) for optimal $p = \frac{1}{\sqrt{\kappa}}$	$\sqrt{\kappa}$ <sup>(h)</sup>	$d$	$\frac{1}{L}$	✓	$\tilde{\mathcal{O}}(\sqrt{\kappa})$ <sup>(c)</sup>	✓

<sup>(a)</sup> This is a special case of S-Local-SVRG, which is a more general method presented in (Gorbunov et al., 2021). S-Local-GD arises as a special case when full gradient is computed on each client.

<sup>(b)</sup> FedLin is a variant with a fixed but different number of local steps for each client. Earlier method S-Local-GD has the same update but random loop length.

<sup>(c)</sup> The  $\tilde{\mathcal{O}}$  notation hides logarithmic factors.

<sup>(d)</sup>  $G$  is the level of dissimilarity from the assumption  $\frac{1}{n} \sum_{i=1}^n \|\nabla f_i(x)\|^2 \leq G^2 + 2LB^2 (f(x) - f_*)$ ,  $\forall x$ .

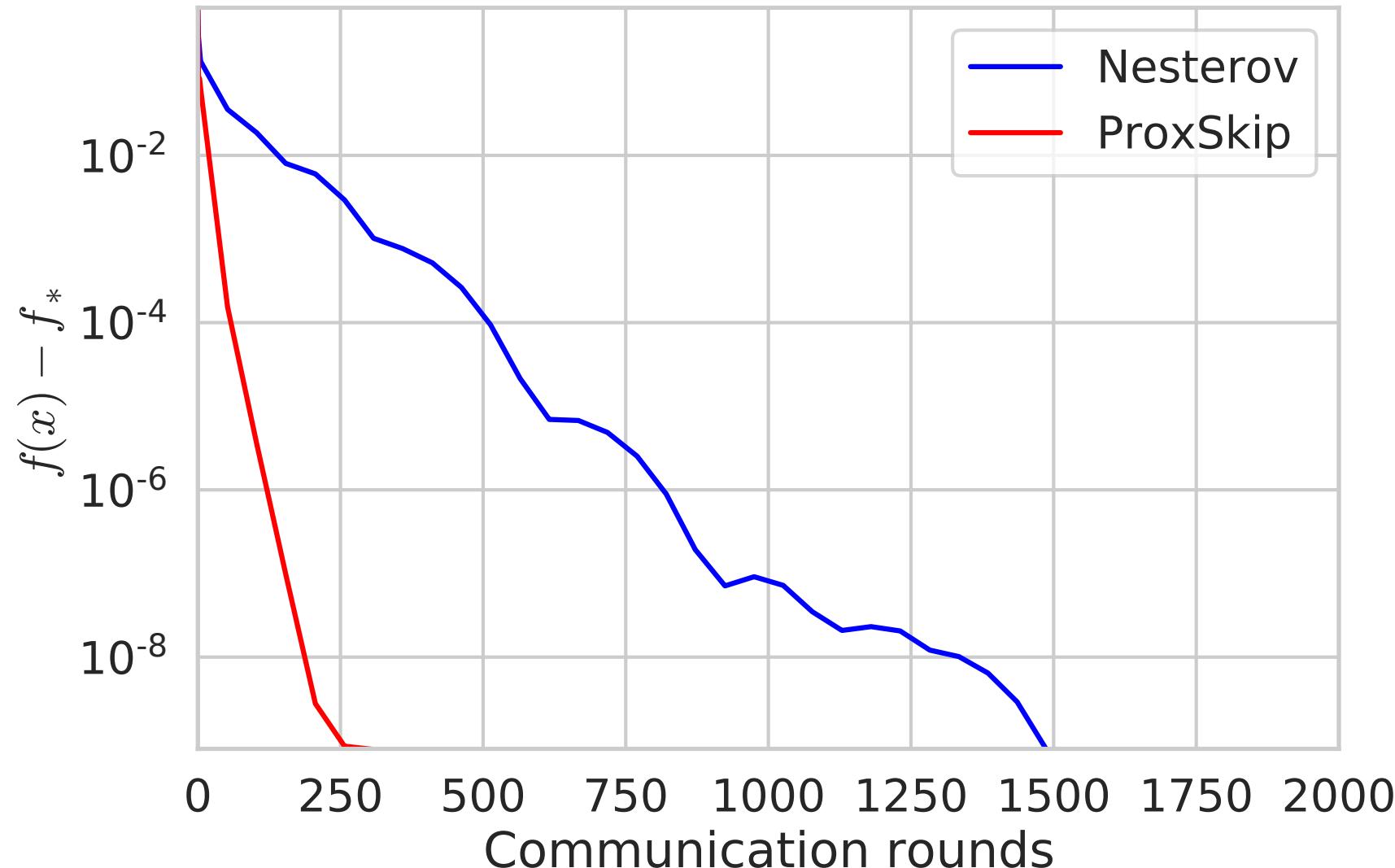
<sup>(e)</sup> We use Scaffold's cumulative local-global stepsize  $\eta_l \eta_g$  for a fair comparison.

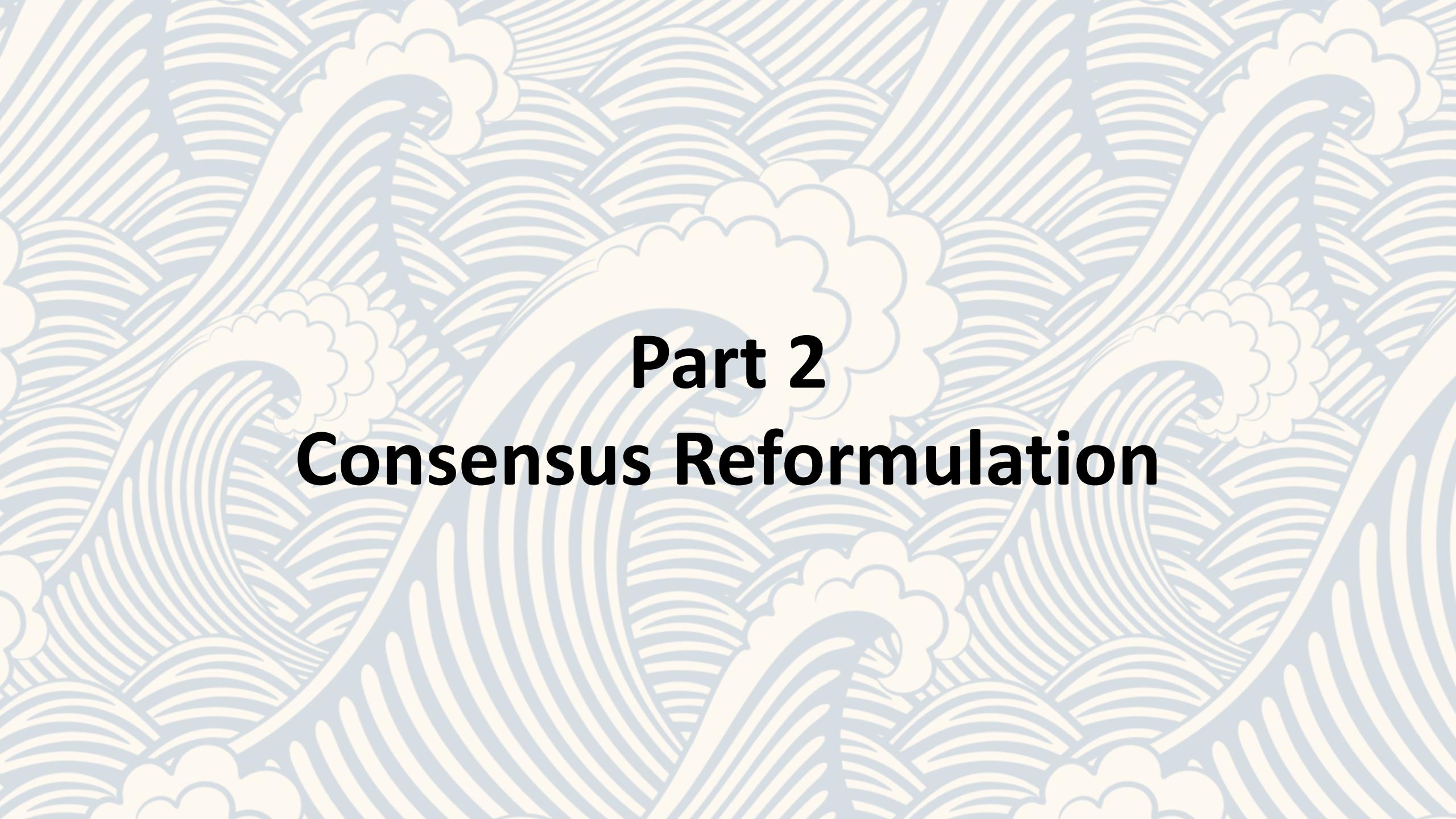
<sup>(f)</sup> The number of sent vectors depends on hyper-parameters, and it is randomized.

<sup>(g)</sup> Scaffnew (Algorithm 2) = ProxSkip (Algorithm 1) applied to the consensus formulation (6) + (7) of the finite-sum problem (5).

<sup>(h)</sup> ProxSkip (resp. Scaffnew) takes a *random* number of gradient (resp. local) steps before prox (resp. communication) is computed (resp. performed). What is shown in the table is the *expected* number of gradient (resp. local) steps.

# Federated Learning: ProxSkip vs Nesterov





## **Part 2**

# **Consensus Reformulation**

# Consensus Reformulation

**Original problem:**  
optimization in  $\mathbb{R}^d$

$$\min_{x \in \mathbb{R}^d} \left\{ f(x) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n f_i(x) \right\}$$

**Bad:** Non-differentiable function

**Consensus reformulation:**  
optimization in  $\mathbb{R}^{nd}$

$$\min_{x_1, \dots, x_n \in \mathbb{R}^d} \left\{ \frac{1}{n} \sum_{i=1}^n f_i(x_i) + \psi(x_1, \dots, x_n) \right\}$$

**Good:** Indicator function of a nonempty closed convex set

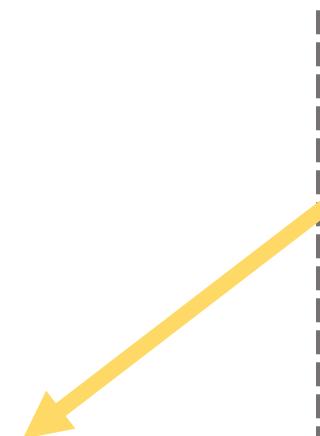
$$\psi(x_1, \dots, x_n) \stackrel{\text{def}}{=} \begin{cases} 0, & \text{if } x_1 = \dots = x_n, \\ +\infty, & \text{otherwise.} \end{cases}$$

# Generalization 1: Constrained Optimization

**Consensus reformulation:**

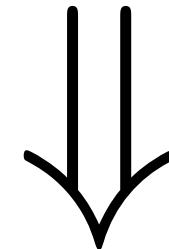
optimization in  $\mathbb{R}^{nd}$

$$\min_{x_1, \dots, x_n \in \mathbb{R}^d} \left\{ \frac{1}{n} \sum_{i=1}^n f_i(x_i) + \psi(x_1, \dots, x_n) \right\}$$



$$\psi(x_1, \dots, x_n) \stackrel{\text{def}}{=} \begin{cases} 0, & \text{if } x_1 = \dots = x_n, \\ +\infty, & \text{otherwise.} \end{cases}$$

**Generalization 1**



$$\psi(x_1, \dots, x_n) \stackrel{\text{def}}{=} \begin{cases} 0, & \text{if } (x_1, \dots, x_n) \in C, \\ +\infty, & \text{otherwise.} \end{cases}$$

Arbitrary closed convex set  
(constraint)



# Generalization 2: Composite Optimization

**Consensus reformulation:**

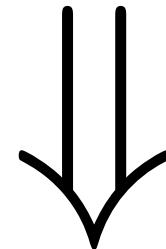
optimization in  $\mathbb{R}^{nd}$

$$\min_{x_1, \dots, x_n \in \mathbb{R}^d} \left\{ \frac{1}{n} \sum_{i=1}^n f_i(x_i) + \psi(x_1, \dots, x_n) \right\}$$



$$\psi(x_1, \dots, x_n) \stackrel{\text{def}}{=} \begin{cases} 0, & \text{if } x_1 = \dots = x_n, \\ +\infty, & \text{otherwise.} \end{cases}$$

**Generalization 2**



$\psi(x_1, \dots, x_n) : \mathbb{R}^{nd} \rightarrow \mathbb{R} \cup \{+\infty\}$   
is a proper closed convex function

The epigraph of  $\psi$  is a closed and convex set

$$\text{epi}(\psi) \stackrel{\text{def}}{=} \{(x, t) \mid \psi(x) \leq t\}$$



# Conceptual Simplification: from $nd$ to $d'$

Composite optimization:

optimization in  $\mathbb{R}^{nd}$

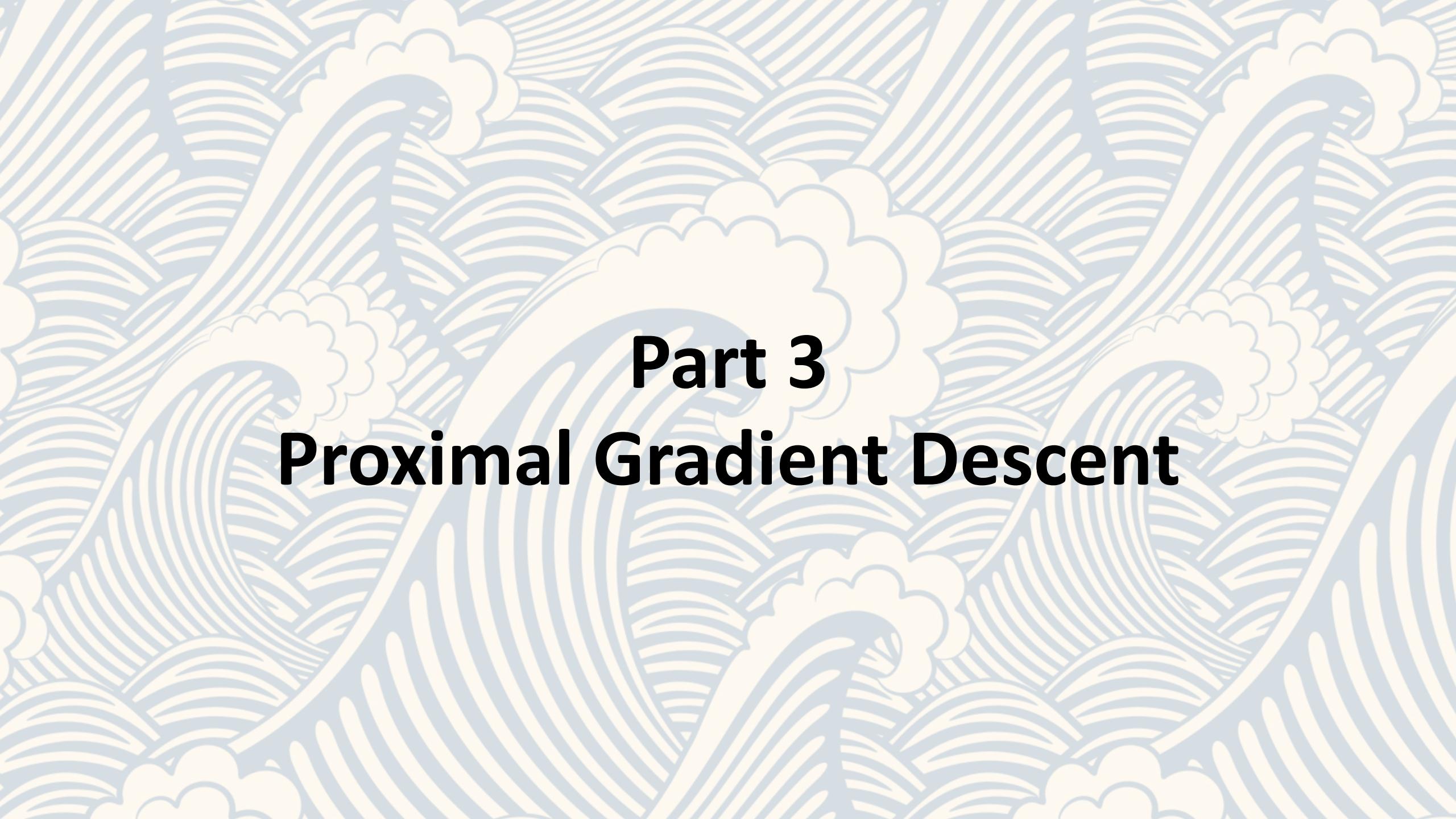
$$\min_{x_1, \dots, x_n \in \mathbb{R}^d} \left\{ \frac{1}{n} \sum_{i=1}^n f_i(x_i) + \psi(x_1, \dots, x_n) \right\}$$

Composite optimization:

optimization in  $\mathbb{R}^{d'}$

$$\min_{x \in \mathbb{R}^{d'}} \{f(x) + \psi(x)\}$$

$$\left\{ \begin{array}{l} d' = nd \\ x = (x_1, \dots, x_n) \\ f(x) = \frac{1}{n} \sum_{i=1}^n f_i(x_i) \\ \psi(x) = \psi(x_1, \dots, x_n) \end{array} \right.$$



# **Part 3**

# **Proximal Gradient Descent**

# Three Assumptions

The epigraph of  $\psi$  is a closed and convex set

$$\text{epi}(\psi) \stackrel{\text{def}}{=} \{(x, t) \in \mathbb{R}^d \times \mathbb{R} \mid \psi(x) \leq t\}$$

$$\min_{x \in \mathbb{R}^d} f(x) + \psi(x)$$

A1  $f$  is  $\mu$ -convex and  $L$ -smooth:

$$\frac{\mu}{2}\|x - y\|^2 \leq D_f(x, y) \leq \frac{L}{2}\|x - y\|^2$$

Bregman divergence of  $f$ :

$$D_f(x, y) \stackrel{\text{def}}{=} f(x) - f(y) - \langle \nabla f(y), x - y \rangle$$

A2  $\psi : \mathbb{R}^d \rightarrow \mathbb{R} \cup \{+\infty\}$  is proper, closed, and convex

A3  $\psi$  is proximable

The proximal operator  $\text{prox}_\psi : \mathbb{R}^d \rightarrow \mathbb{R}^d$  defined by

$$\text{prox}_\psi(x) \stackrel{\text{def}}{=} \arg \min_{u \in \mathbb{R}^d} \left( \psi(u) + \frac{1}{2}\|u - x\|^2 \right)$$

can be evaluated exactly (e.g., in closed form)

# Key Method: Proximal Gradient Descent

proximal operator:

$$\text{prox}_\psi(x) \stackrel{\text{def}}{=} \arg \min_{u \in \mathbb{R}^d} \left( \psi(u) + \frac{1}{2} \|u - x\|^2 \right)$$

$$x_t - \gamma \nabla f(x_t)$$

stepsize

gradient operator

$$x \mapsto x - \gamma \nabla f(x)$$

# Proximal Gradient Descent: Theory

Theorem:

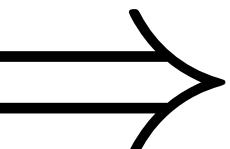
$f$  is  $\mu$ -convex and  $L$ -smooth:  
 $\frac{\mu}{2}\|x - y\|^2 \leq D_f(x, y) \leq \frac{L}{2}\|x - y\|^2$   
 $\frac{L}{\mu}$  is the condition number of  $f$

$$t \geq \frac{L}{\mu} \log \frac{1}{\varepsilon} \quad \Rightarrow \quad \|x_t - x_\star\|^2 \leq \varepsilon \|x_0 - x_\star\|^2$$

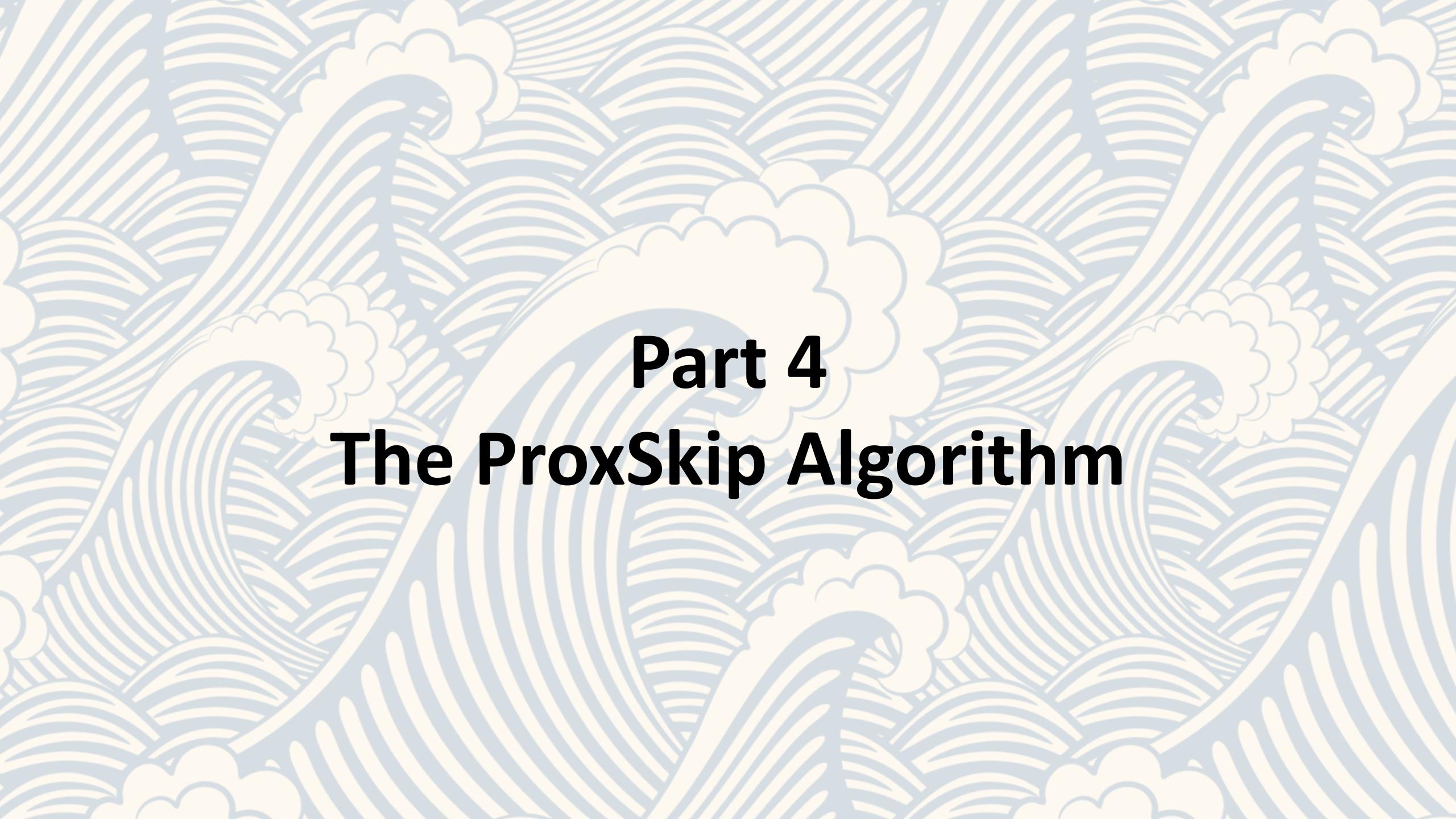
# iterations

Error tolerance

$$x_\star \stackrel{\text{def}}{=} \arg \min_{x \in \mathbb{R}^d} f(x) + \psi(x)$$



(for stepsize  $\gamma = \frac{1}{L}$ )



# **Part 4**

# **The ProxSkip Algorithm**

# What to do When the Prox is Expensive?

Can we somehow get away with  
fewer evaluations of the proximity operator  
in the Proximal GD method?

## Approach 1



We'll skip ALL prox evaluations!



The method is NOT implementable!



Serves as an inspiration for Approach 2

## Approach 2 (ProxSkip)



We'll skip MANY prox evaluations!



The method is implementable!

**Approach 1:  
Simple, Extreme but  
Practically Useless Variant**

# Removing $\psi$ via a Reformulation

$$\min_{x \in \mathbb{R}^d} f(x) - \langle h_\star, x \rangle$$

$$h_\star \stackrel{\text{def}}{=} \nabla f(x_\star)$$

$$x_\star \stackrel{\text{def}}{=} \arg \min_{x \in \mathbb{R}^d} f(x) + \psi(x)$$



$x_\star$  is a solution of the above problem!

By the 1st order optimality conditions, the solution satisfies  $\nabla f(x) - \nabla f(x_\star) = 0$



We do not know  $h_\star = \nabla f(x_\star)$ !

# Apply Gradient Descent to the Reformulation

$$\begin{aligned} h_\star &\stackrel{\text{def}}{=} \nabla f(x_\star) \\ x_\star &\stackrel{\text{def}}{=} \arg \min_{x \in \mathbb{R}^d} f(x) + \psi(x) \end{aligned}$$

$$x_{t+1} = x_t - \gamma (\nabla f(x_t) - h_\star)$$



We do not need to evaluate the prox of  $\psi$  at all!



We do not know  $h_\star$  and hence can't implement the method!

# Idea: Try to “Learn” the Optimal Gradient Shift

$$x_{t+1} = x_t - \gamma (\nabla f(x_t) - h_t)$$

Desire:  $h_t \rightarrow h_*$



Perhaps we can learn  $h_*$  with only occasional access to  $\psi$ ?

# **Approach 2: The ProxSkip Method**

# ProxSkip: The Algorithm (Bird's Eye View)

1

$$\hat{x}_{t+1} = x_t - \gamma (\nabla f(x_t) - h_t)$$

2a

with probability  $1 - p$  do

$$1 - p \approx 1$$

$$x_{t+1} = \hat{x}_{t+1}$$

$$h_{t+1} = h_t$$

2b

with probability  $p$  do

$$p \approx 0$$

evaluate  $\text{prox}_{\frac{\gamma}{p}\psi}(?)$

$$x_{t+1} = ?$$

$$h_{t+1} = ?$$

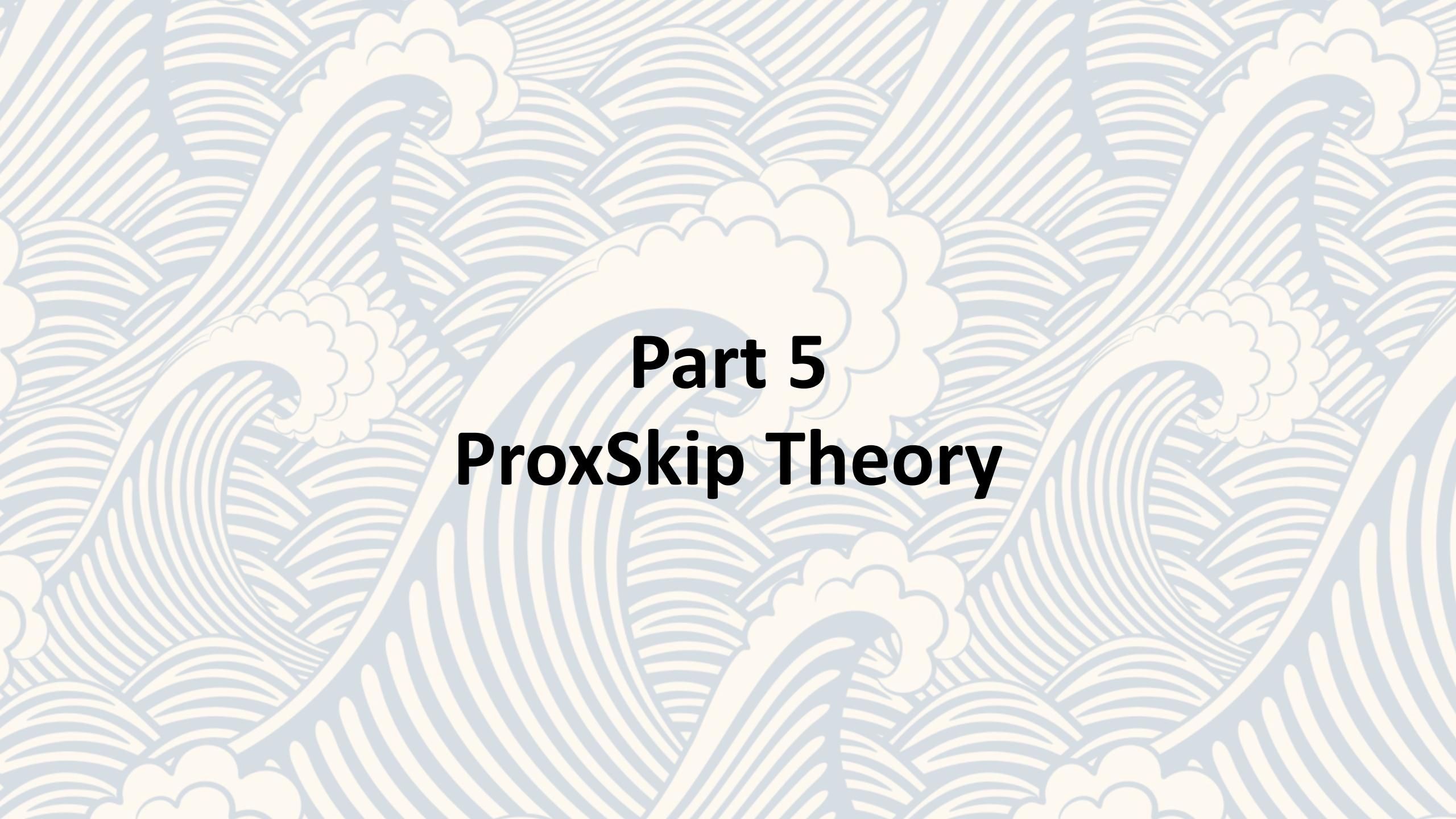
# ProxSkip: The Algorithm (Detailed View)

---

## Algorithm 1 ProxSkip

```
1: stepsize  $\gamma > 0$ , probability  $p > 0$ , initial iterate  $x_0 \in \mathbb{R}^d$ , initial control variate  $\mathbf{h}_0 \in \mathbb{R}^d$ , number of iterations  $T \geq 1$ 
2: for  $t = 0, 1, \dots, T - 1$  do
3:    $\hat{x}_{t+1} = x_t - \gamma(\nabla f(x_t) - \mathbf{h}_t)$            ◊ Take a gradient-type step adjusted via the control variate  $\mathbf{h}_t$ 
4:   Flip a coin  $\theta_t \in \{0, 1\}$  where  $\text{Prob}(\theta_t = 1) = p$       ◊ Flip a coin that decides whether to skip the prox or not
5:   if  $\theta_t = 1$  then
6:      $x_{t+1} = \text{prox}_{\frac{\gamma}{p}\psi}(\hat{x}_{t+1} - \frac{\gamma}{p}\mathbf{h}_t)$           ◊ Apply prox, but only very rarely! (with small probability  $p$ )
7:   else
8:      $x_{t+1} = \hat{x}_{t+1}$                                               ◊ Skip the prox!
9:   end if
10:   $\mathbf{h}_{t+1} = \mathbf{h}_t + \frac{p}{\gamma}(x_{t+1} - \hat{x}_{t+1})$           ◊ Update the control variate  $\mathbf{h}_t$ 
11: end for
```

---



# **Part 5**

# **ProxSkip Theory**

# ProxSkip: Bounding the # of Iterations

**Theorem:**

$f$  is  $\mu$ -convex and  $L$ -smooth:  
 $\frac{\mu}{2}\|x - y\|^2 \leq D_f(x, y) \leq \frac{L}{2}\|x - y\|^2$   
 $\frac{L}{\mu}$  is the condition number of  $f$

$$t \geq \max \left\{ \frac{L}{\mu}, \frac{1}{p^2} \right\} \log \frac{1}{\varepsilon} \quad \Rightarrow \quad \mathbb{E} [\Psi_t] \leq \varepsilon \Psi_0$$

# iterations

$p$  = probability of evaluating the prox

Lyapunov function:

$$\Psi_t \stackrel{\text{def}}{=} \|x_t - x_\star\|^2 + \frac{1}{L^2 p^2} \|h_t - h_\star\|^2$$

# ProxSkip: Optimal Prox-Evaluation Probability

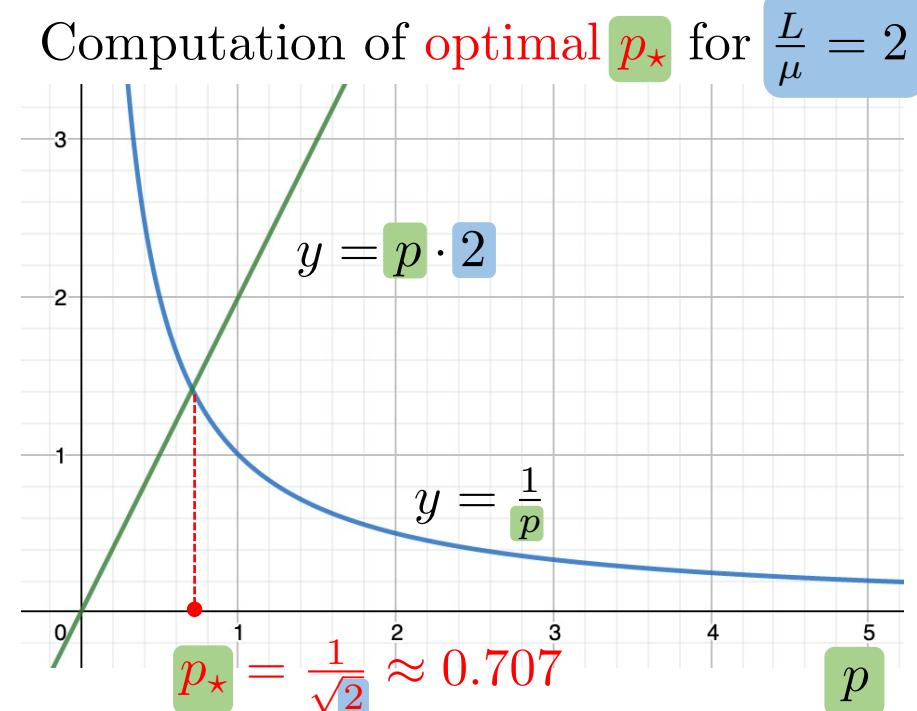
Since in each iteration we evaluate the prox with probability  $p$ ,  
the expected number of prox evaluations after  $t$  iterations is:

$$p \cdot t = p \cdot \max \left\{ \frac{L}{\mu}, \frac{1}{p^2} \right\} \cdot \log \frac{1}{\varepsilon} = \max \left\{ p \cdot \frac{L}{\mu}, \frac{1}{p} \right\} \cdot \log \frac{1}{\varepsilon}$$

$\frac{L}{\mu}$  is the condition number of  $f$

Minimized for  $p$  satisfying  $p \cdot \frac{L}{\mu} = \frac{1}{p}$

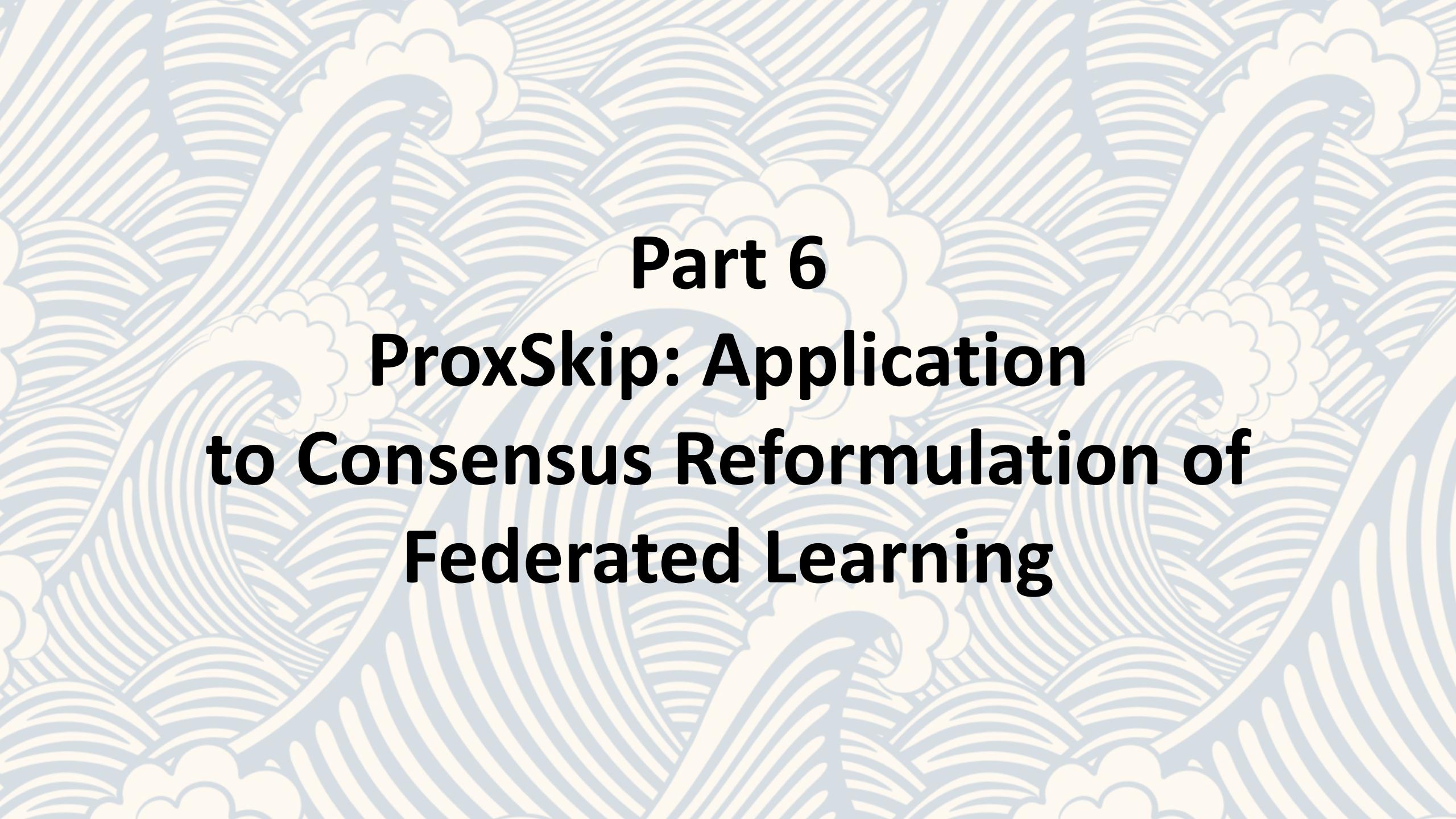
$$\Rightarrow p_{\star} = \frac{1}{\sqrt{L/\mu}}$$



# ProxSkip: # of Gradient and Prox Evaluations

$$p_\star = \frac{1}{\sqrt{L/\mu}} \Rightarrow$$

# of iterations	$\max \left\{ \frac{L}{\mu}, \frac{1}{p^2} \right\} \cdot \log \frac{1}{\varepsilon}$	$\frac{L}{\mu} \cdot \log \frac{1}{\varepsilon}$
# of gradient evaluations	$\max \left\{ \frac{L}{\mu}, \frac{1}{p^2} \right\} \cdot \log \frac{1}{\varepsilon}$	$\frac{L}{\mu} \cdot \log \frac{1}{\varepsilon}$
Expected # of prox evaluations	$\max \left\{ p \cdot \frac{L}{\mu}, \frac{1}{p} \right\} \cdot \log \frac{1}{\varepsilon}$	$\sqrt{\frac{L}{\mu}} \cdot \log \frac{1}{\varepsilon}$
Expected # of gradient evaluations between 2 prox evaluations	$\frac{1}{p}$	$\sqrt{\frac{L}{\mu}}$



# **Part 6**

# **ProxSkip: Application to Consensus Reformulation of Federated Learning**

# Federated Learning: ProxSkip vs Baselines

Table 1. The performance of federated learning methods employing multiple local gradient steps in the strongly convex regime.

method	# local steps per round	# floats sent per round	stepsize on client $i$	linear rate?	# rounds	rate better than GD?
GD (Nesterov, 2004)	1	$d$	$\frac{1}{L}$	✓	$\tilde{\mathcal{O}}(\kappa)$ <sup>(c)</sup>	✗
LocalGD (Khaled et al., 2019; 2020)	$\tau$	$d$	$\frac{1}{\tau L}$	✗	$\mathcal{O}\left(\frac{G^2}{\mu n \tau \varepsilon}\right)$ <sup>(d)</sup>	✗
Scaffold (Karimireddy et al., 2020)	$\tau$	$2d$	$\frac{1}{\tau L}$ <sup>(e)</sup>	✓	$\tilde{\mathcal{O}}(\kappa)$ <sup>(c)</sup>	✗
S-Local-GD <sup>(a)</sup> (Gorbunov et al., 2021)	$\tau$	$d < \# < 2d$ <sup>(f)</sup>	$\frac{1}{\tau L}$	✓	$\tilde{\mathcal{O}}(\kappa)$	✗
FedLin <sup>(b)</sup> (Mitra et al., 2021)	$\tau_i$	$2d$	$\frac{1}{\tau_i L}$	✓	$\tilde{\mathcal{O}}(\kappa)$ <sup>(c)</sup>	✗
Scaffnew <sup>(g)</sup> (this work) for any $p \in (0, 1]$	$\frac{1}{p}$ <sup>(h)</sup>	$d$	$\frac{1}{L}$	✓	$\tilde{\mathcal{O}}\left(p\kappa + \frac{1}{p}\right)$ <sup>(c)</sup>	✓ (for $p > \frac{1}{\kappa}$ )
Scaffnew <sup>(g)</sup> (this work) for optimal $p = \frac{1}{\sqrt{\kappa}}$	$\sqrt{\kappa}$ <sup>(h)</sup>	$d$	$\frac{1}{L}$	✓	$\tilde{\mathcal{O}}(\sqrt{\kappa})$ <sup>(c)</sup>	✓

<sup>(a)</sup> This is a special case of S-Local-SVRG, which is a more general method presented in (Gorbunov et al., 2021). S-Local-GD arises as a special case when full gradient is computed on each client.

<sup>(b)</sup> FedLin is a variant with a fixed but different number of local steps for each client. Earlier method S-Local-GD has the same update but random loop length.

<sup>(c)</sup> The  $\tilde{\mathcal{O}}$  notation hides logarithmic factors.

<sup>(d)</sup>  $G$  is the level of dissimilarity from the assumption  $\frac{1}{n} \sum_{i=1}^n \|\nabla f_i(x)\|^2 \leq G^2 + 2LB^2 (f(x) - f_*)$ ,  $\forall x$ .

<sup>(e)</sup> We use Scaffold's cumulative local-global stepsize  $\eta_l \eta_g$  for a fair comparison.

<sup>(f)</sup> The number of sent vectors depends on hyper-parameters, and it is randomized.

<sup>(g)</sup> Scaffnew (Algorithm 2) = ProxSkip (Algorithm 1) applied to the consensus formulation (6) + (7) of the finite-sum problem (5).

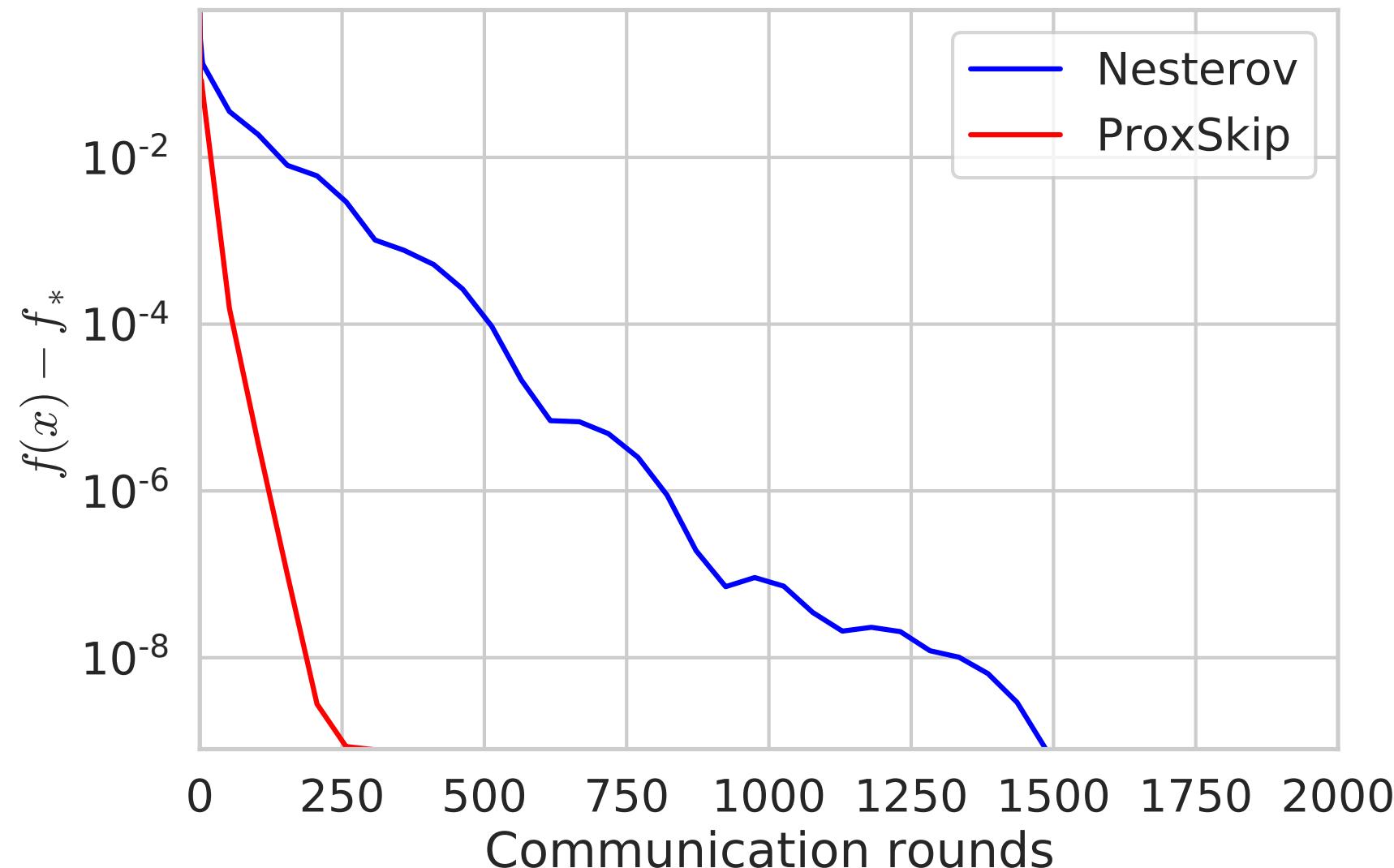
<sup>(h)</sup> ProxSkip (resp. Scaffnew) takes a *random* number of gradient (resp. local) steps before prox (resp. communication) is computed (resp. performed). What is shown in the table is the *expected* number of gradient (resp. local) steps.



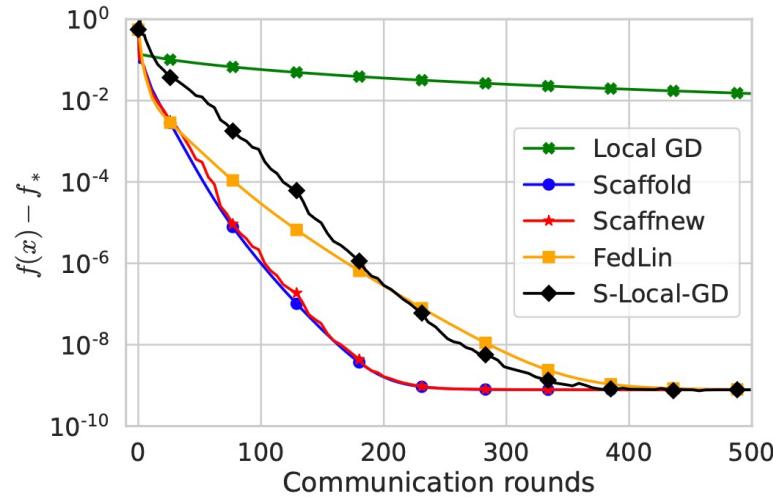
# **Part 7**

# **Experiments**

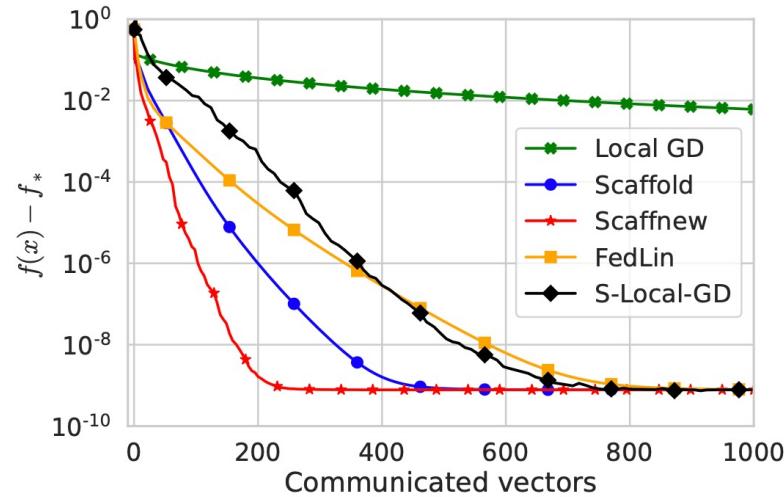
# Scaffnew (=ProxSkip applied to FL) vs Nesterov



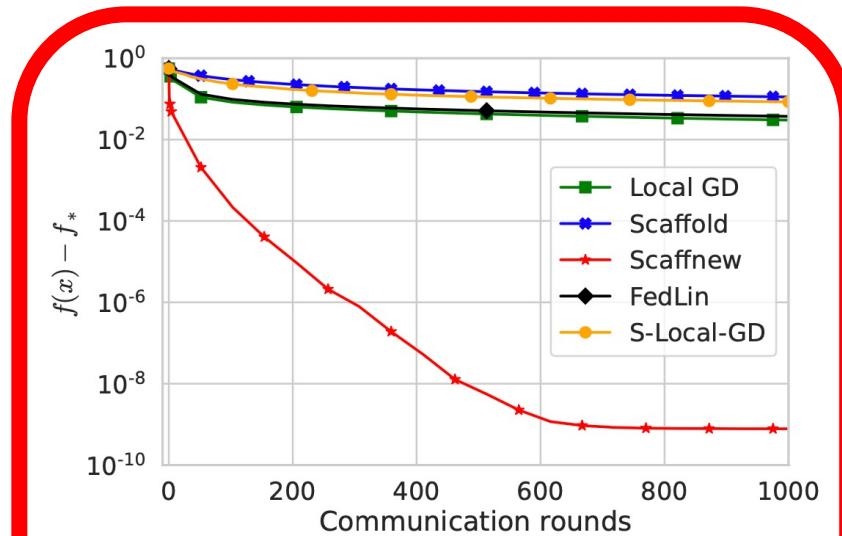
# Scaffnew (=ProxSkip applied to FL) vs Baselines



(a) tuned hyper-parameters



(b) tuned hyper-parameters



(c) theoretical hyper-parameters

**Figure 1. Deterministic Problem.** Comparison of **Scaffnew** to other local update methods that tackle data-heterogeneity and to **LocalGD**. In (a) we compare communication rounds with optimally tuned hyper-parameters. In (b) we compare communicated vectors (**Scaffold**, **FedLin** and **S-Local-GD** require transmission of additional variables). In (c), we compare communication rounds with the algorithm parameters set to the best theoretical stepsizes used in the convergence proofs.

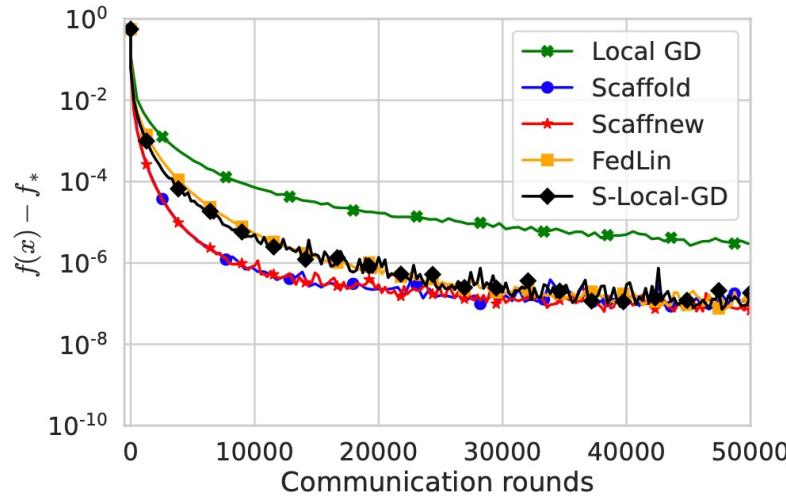
L2-regularized logistic regression:

$$f(x) = \frac{1}{n} \sum_{i=1}^n \log (1 + \exp (-b_i a_i^\top x)) + \frac{\lambda}{2} \|x\|^2$$

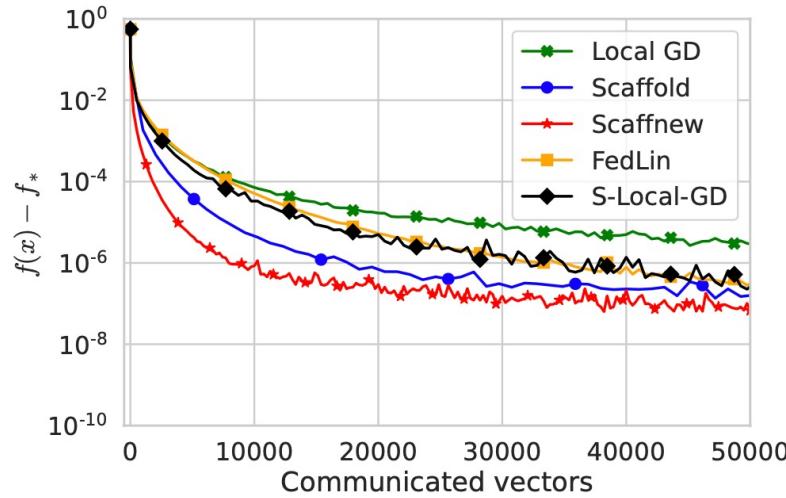
$$a_i \in \mathbb{R}^d, b_i \in \{-1, +1\}, \lambda = L/10^4$$

w8a dataset from LIBSVM library (Chang & Lin, 2011)

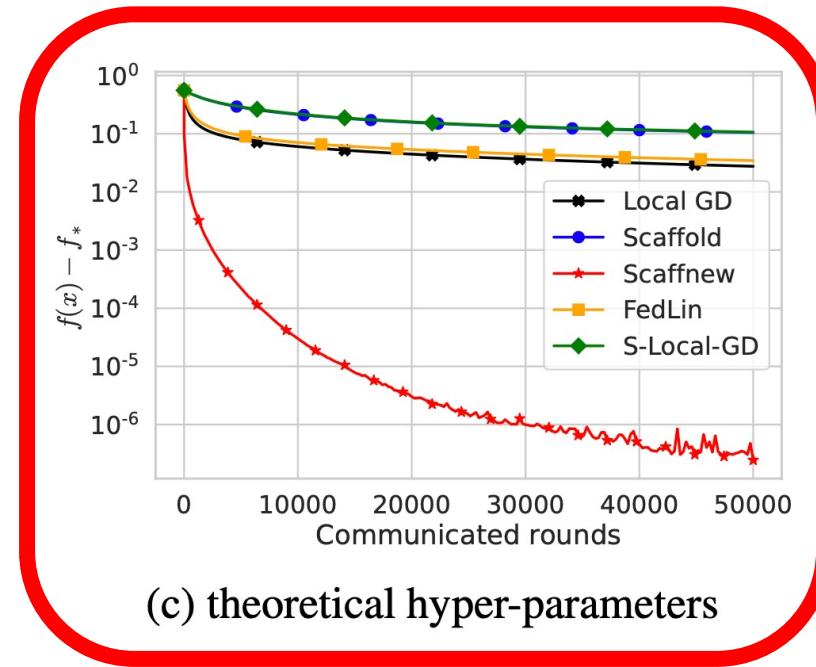
# Scaffnew (=ProxSkip applied to FL) vs Baselines



(a) tuned hyper-parameters



(b) tuned hyper-parameters



(c) theoretical hyper-parameters

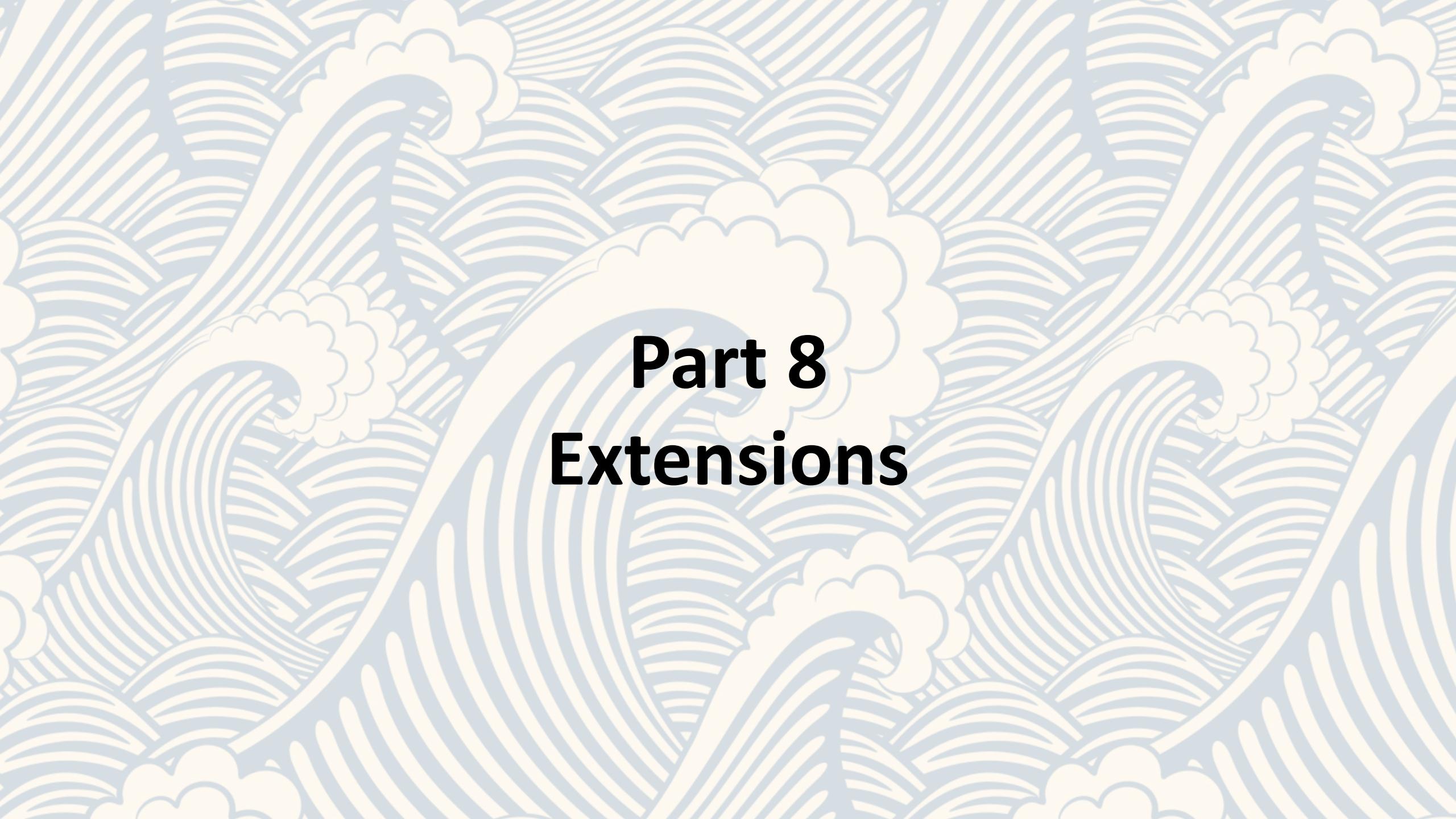
**Figure 2. Stochastic Problem.** Comparison of **Scaffnew** to other local update methods that tackle data-heterogeneity and to **LocalSGD**. In (a) we compare communication rounds with optimally tuned hyper-parameters. In (b) we compare communicated vectors and in (c), we compare communication rounds with the algorithm parameters set to the best theoretical stepsizes used in the convergence proofs.

L2-regularized logistic regression:

$$f(x) = \frac{1}{n} \sum_{i=1}^n \log(1 + \exp(-b_i a_i^\top x)) + \frac{\lambda}{2} \|x\|^2$$

$$a_i \in \mathbb{R}^d, b_i \in \{-1, +1\}, \lambda = L/10^4$$

w8a dataset from LIBSVM library (Chang & Lin, 2011)



# **Part 8**

# **Extensions**

# Extension 1: From Gradients to Stochastic Gradients

- As described, in ProxSkip each worker computes the **full gradient** of its local function
- It's often better to consider a **cheap stochastic approximation of the gradient** instead
  - We consider this extension in the paper
  - We provide theoretical convergence rates

$$\nabla f_i(x_t) \Rightarrow g_i(x_t)$$

Full gradient                              Stochastic gradient

**Assumptions:**

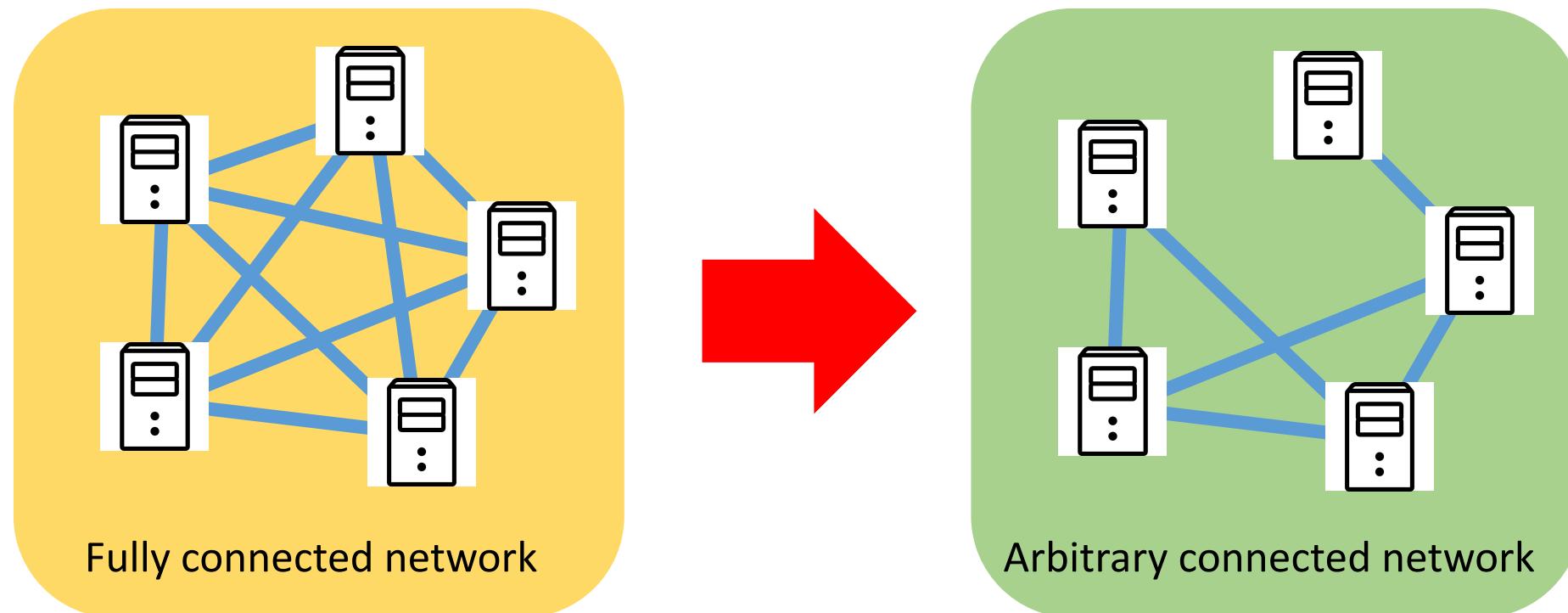
**(expected smoothness)**  $E \left[ \|g_{i,t}(x_t) - \nabla f(x_*)\|^2 \mid x_t \right] \leq 2AD_f(x_t, x_*) + C$

(Gower et al, 2019)

**(unbiasedness)**  $\mathbb{E} [g_{i,t}(x_t) \mid x_t] = \nabla f_i(x_t)$

# Extension 2: From Fully Connected Network to Arbitrary Connected Network

- In each communication round of ProxSkip, **each worker sends messages to all other workers** (e.g., through a server).
  - We can think of ProxSkip workers as the nodes of a **fully-connected network**.
  - In each communication round, all **workers communicate with their neighbors**.
- In the paper we provide extension to **arbitrary connected networks**.





**The End**