



# Optimization for Deep Learning

## Lecture 13-3: Decentralized Deterministic Optimization

Kun Yuan

# Part 1

---

## Decentralized gradient descent (DGD)

# Distributed optimization

---

- A network of  $n$  nodes collaborate to solve the following deterministic optimization problem:

$$\min_{x \in \mathbb{R}^d} \sum_{i=1}^n f_i(x)$$

We assume **data heterogeneity** exists, i.e.,  $f_i(x) \neq f_j(x)$

- Given a connected network, decentralized gradient descent iterates as follows [NO09]

$$x_i^{(k+1)} = \sum_{j \in \mathcal{N}_i} w_{ij} x_j^{(k)} - \gamma \nabla f_i(x_i^{(k)}) \quad \forall i \in [n] \quad (\text{adapt-with-combine})$$

A different adapt-then-combine version was also proposed in [LS07]

$$x_i^{(k+1)} = \sum_{j \in \mathcal{N}_i} w_{ij} (x_j^{(k)} - \gamma \nabla f_j(x_j^{(k)})) \quad \forall i \in [n] \quad (\text{adapt-then-combine})$$

# Decentralized gradient descent (DGD)

---

- Both adapt-with-combine and adapt-then-combine versions are used a lot
- Adapt-with-combine version is easier to analyze while adapt-then-combine performs better in real implementations
- For simplicity, we mainly discuss DGD in the adapt-with-combine version
- We introduce the following notations for convenience

$$\mathbf{x} = [x_1^T; x_2^T; \dots; x_n^T] \in \mathbb{R}^{n \times d}$$

$$\nabla \mathbf{F}(\mathbf{x}) = [\nabla f_1(x_1)^T; \nabla f_2(x_2)^T; \dots; \nabla f_n(x_n)^T] \in \mathbb{R}^{n \times d}$$

# Decentralized gradient descent (DGD)

- With the notations, DGD can be rewritten in a neat manner

$$\mathbf{x}^{(k+1)} = W\mathbf{x}^k - \gamma \nabla \mathbf{F}(\mathbf{x}^{(k)})$$

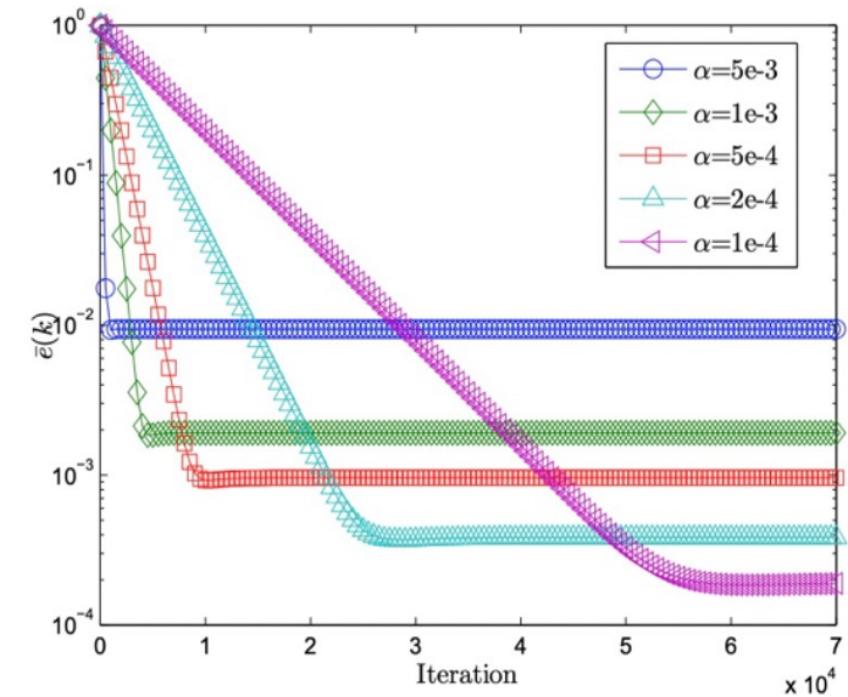
- Very intuitive, but **does not exactly converge** to the desired solution with constant learning rate

## Theorem 2 [DGD Convergence] [YLY16]

Assume each  $f_i(x)$  is  $L$ -smooth and  $\mu$ -strongly convex.  
 If the learning rate is set as  $\gamma = O(1/L)$ , DGD converges as follows

$$\frac{1}{n} \sum_{i=1}^n \|x_i^{(k)} - x^*\| = O\left(\rho^k + \frac{\gamma b}{1-\rho}\right)$$

where  $\rho \in (0, 1)$  and  $b^2 = \frac{1}{n} \sum_{i=1}^n \|\nabla f_i(x^*)\|^2$ .



# DGD is essentially a GD algorithm

- DGD was proposed by heuristics, and it cannot converge to the desired solution
- Why DGD cannot converge to the desired solution? Can we correct the bias?
- We need to treat decentralized optimization in a more “**scientific**” manner

## Lemma 2

Assume matrix  $W$  is symmetric, primitive, and doubly stochastic, we have

$$Wx = x \iff x_1 = x_2 = \dots = x_n$$

# DGD is essentially a GD algorithm

---

- The distributed optimization problem can be reformulated into a constrained problem

$$\min_{x \in \mathbb{R}^d} \sum_{i=1}^n f_i(x_i) \quad \text{s.t.} \quad x_1 = x_2 = \dots = x_n \quad (1)$$

where the constraint  $x_1 = x_2 = \dots = x_n$  is typically referred to as **consensus constraint**

- With Lemma 2, the above constrained problem is equivalent to [YLY16]

$$\min_{\mathbf{x} \in \mathbb{R}^{n \times d}} \mathbf{F}(\mathbf{x}) \quad \text{s.t.} \quad (I - W)\mathbf{x} = 0 \quad (2)$$

where  $\mathbf{F}(\mathbf{x}) = \sum_{i=1}^n f_i(x_i)$

# DGD is essentially a GD algorithm

---

- Since  $W$  is symmetric, problem (2) is further equivalent to

$$\min_{\mathbf{x} \in \mathbb{R}^{n \times d}} \quad \mathbf{F}(\mathbf{x}) \quad \text{s.t.} \quad (I - W)^{1/2} \mathbf{x} = 0$$

- To solve the above constrained problem, we penalize the constraint to the objective function

$$\min_{\mathbf{x} \in \mathbb{R}^{n \times d}} \quad \mathbf{F}(\mathbf{x}) + \frac{1}{2\gamma} \|(I - W)^{1/2} \mathbf{x}\|^2 \quad (\text{penalized problem})$$

- Solve the above problem with gradient descent

$$\begin{aligned} \mathbf{x}^{(k+1)} &= \mathbf{x}^{(k)} - \gamma \nabla_{\mathbf{x}} \left( \mathbf{F}(\mathbf{x}) + \frac{1}{2\gamma} \|(I - W)^{1/2} \mathbf{x}\|^2 \right) \\ &= \mathbf{x}^{(k)} - \gamma \nabla F(\mathbf{x}^{(k)}) - (I - W)\mathbf{x}^{(k)} \\ &= W\mathbf{x}^{(k)} - \gamma \nabla F(\mathbf{x}^{(k)}) \end{aligned}$$

We recovered DGD!

# DGD is essentially a GD algorithm

---

- The DGD algorithm targeting to solve the consensus-constrained problem

$$\min_{x \in \mathbb{R}^d} \quad \sum_{i=1}^n f_i(x_i) \quad \text{s.t.} \quad x_1 = x_2 = \dots = x_n$$

is essentially a gradient descent algorithm that solves an approximate penalized problem

$$\min_{\mathbf{x} \in \mathbb{R}^{n \times d}} \quad \mathbf{F}(\mathbf{x}) + \frac{1}{2\gamma} \|(I - W)^{1/2}\mathbf{x}\|^2$$

- Since DGD does not solve the original problem, it cannot converge to the desired solution



## Part 2

---

### An Exact First-Order algorithm (EXTRA)

# EXTRA algorithm

---

- Recall the consensus-constrained optimization problem

$$\min_{\mathbf{x} \in \mathbb{R}^{n \times d}} \quad \mathbf{F}(\mathbf{x}) \quad \text{s.t.} \quad \left(\frac{\mathbf{I} - \mathbf{W}}{2}\right)^{1/2} \mathbf{x} = 0$$

- When solve the above problem with penalty method, we reduce to DGD; cannot converge exactly
- Many effective algorithms to handle the constraints, such as Augmented Lagrangian Method
- The Augmented Lagrangian function is

$$L(\mathbf{x}, \mathbf{y}) = \mathbf{F}(\mathbf{x}) + \frac{1}{\gamma} \langle \mathbf{y}, \left(\frac{\mathbf{I} - \mathbf{W}}{2}\right)^{1/2} \mathbf{x} \rangle + \frac{1}{2\gamma} \left\| \left(\frac{\mathbf{I} - \mathbf{W}}{2}\right)^{1/2} \mathbf{x} \right\|^2$$

# EXTRA algorithm

---

- The ALM algorithm to solve the constrained problem is

$$\begin{aligned}\mathbf{x}^{(k+1)} &= \mathbf{x}^{(k)} - \gamma \nabla_{\mathbf{x}} L(\mathbf{x}^{(k)}, \mathbf{y}^{(k)}) \\ \mathbf{y}^{(k+1)} &= \mathbf{y}^{(k)} + \gamma \nabla_{\mathbf{x}} L(\mathbf{x}^{(k+1)}, \mathbf{y}^{(k)})\end{aligned}$$

which can be expanded as

$$\begin{aligned}\mathbf{x}^{(k+1)} &= \left(\frac{I + W}{2}\right) \mathbf{x}^{(k)} - \gamma \nabla \mathbf{F}(\mathbf{x}^{(k)}) - \left(\frac{I - W}{2}\right)^{1/2} \mathbf{y}^{(k)} \\ \mathbf{y}^{(k+1)} &= \mathbf{y}^{(k)} + \left(\frac{I - W}{2}\right)^{1/2} \mathbf{x}^{(k+1)}\end{aligned}$$

- If we remove the dual variable  $\mathbf{y}$  from the above recursion, we achieve EXTRA [SLWY15]

$$\mathbf{x}^{(k+1)} = \left(\frac{W + I}{2}\right) (2\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}) - \gamma (\nabla \mathbf{F}(\mathbf{x}^{(k)}) - \nabla \mathbf{F}(\mathbf{x}^{(k-1)}))$$

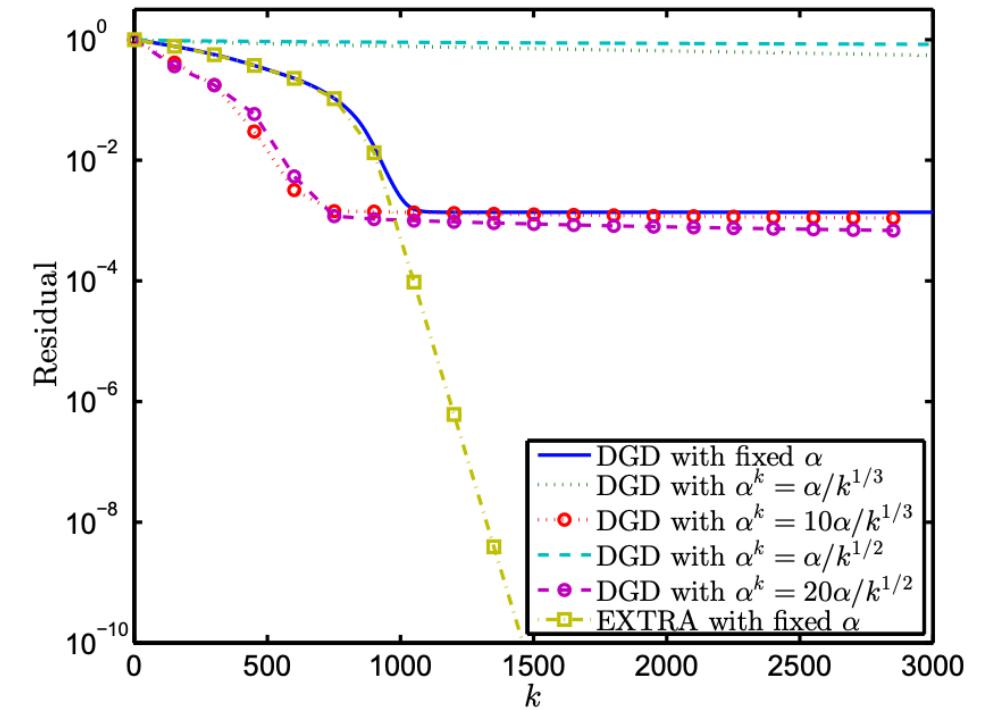
# EXTRA algorithm

- EXTRA can converge to the exact solution

## Theorem 3 [EXTRA Convergence] [SKWY15]

Assume each  $f_i(x)$  is  $L$ -smooth and  $\mu$ -strongly convex. If the learning rate is set as  $\gamma = O(1/L)$ , EXTRA converges as follows

$$\frac{1}{n} \sum_{i=1}^n \|x_i^{(k)} - x^*\| = O(\rho^k)$$



# Exact-Diffusion

- Exact-Diffusion is essentially the adapt-then-combine version of EXTRA

$$\mathbf{x}^{(k+1)} = \left(\frac{W + I}{2}\right) \left[ 2\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)} - \gamma (\nabla \mathbf{F}(\mathbf{x}^{(k)}) - \nabla \mathbf{F}(\mathbf{x}^{(k-1)})) \right]$$

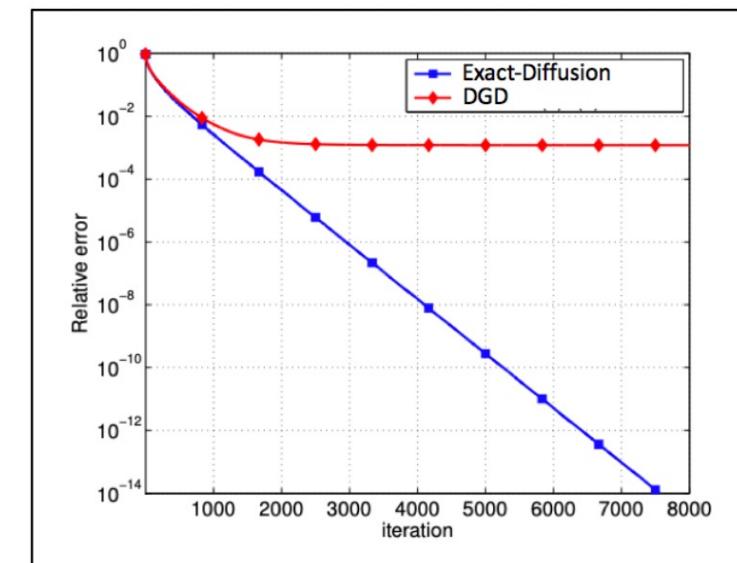
- Exact-Diffusion can be rewritten in a very elegant recursion

**DGD**

$$\begin{aligned} \psi_i^{(k)} &= x_i^{(k)} - \gamma \nabla f_i(x_i^{(k)}) && \text{(compt.)} \\ x_i^{(k+1)} &= \sum_{j \in \mathcal{N}_i} w_{ij} \psi_j^{(k)} && \text{(comm.)} \end{aligned}$$

**Exact Diffusion**

$$\begin{aligned} \psi_i^{(k)} &= x_i^{(k)} - \gamma \nabla f_i(x_i^{(k)}) && \text{(compt.)} \\ \phi_i^{(k)} &= \psi_i^{(k)} + x_i^{(k)} - \psi_i^{(k-1)} && \text{(corret.)} \\ x_i^{(k+1)} &= \sum_{j \in \mathcal{N}_i} w_{ij} \phi_j^{(k)} && \text{(comm.)} \end{aligned}$$





## Part 2

---

### Gradient tracking

- EXTRA and Exact-Diffusion can correct the bias in DGD, but have to use symmetric weight matrix
- In real implementations, the graph may not always be symmetric and it can even be time-varying
- **Gradient tracking** is a new state-of-the-art algorithm in decentralized optimization

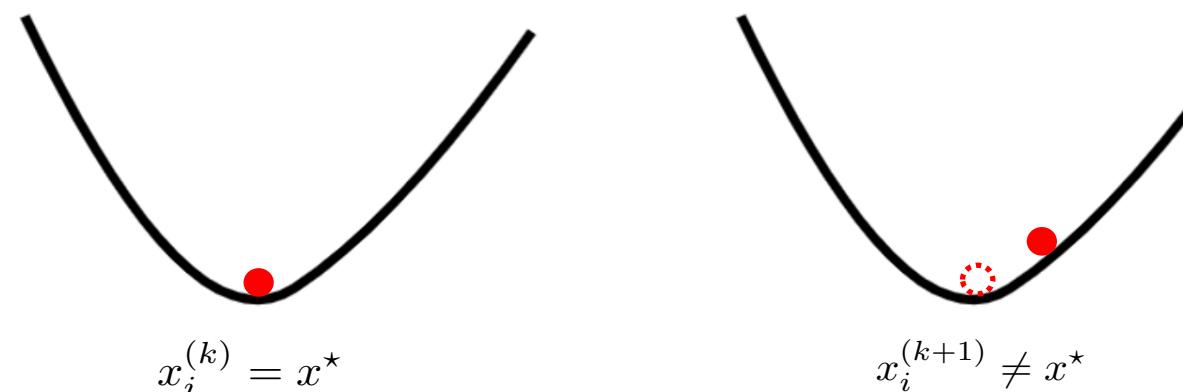
- We revisit the vanilla DGD to figure out the intuition why it does not exactly converge

$$x_i^{(k+1)} = \sum_{j \in \mathcal{N}_i} w_{ij} x_j^{(k)} - \nabla f_i(x_i^{(k)})$$

- Assume all nodes initialize at  $x_i^{(k)} = x^*$ , we examine whether  $x_i^{(k+1)}$  remains at  $x^*$

$$x_i^{(k+1)} = \sum_{j \in \mathcal{N}_i} w_{ij} x^* - \nabla f_i(x^*) = x^* - \nabla f_i(x^*) \neq x^*$$

where the last equality does not hold due to  $f_i(x) \neq f_j(x)$  and hence  $\nabla f_i(x^*) \neq \frac{1}{n} \sum_{i=1}^n \nabla f_i(x^*) = 0$



# A crude idea

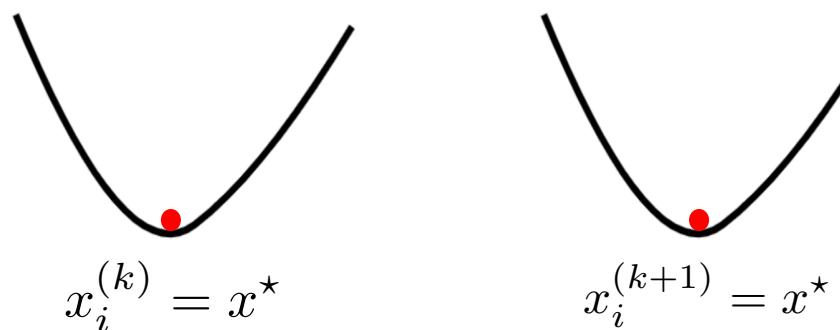
---

- To remove the bias, an impractical but intuitive algorithm is

$$x_i^{(k+1)} = \sum_{j \in \mathcal{N}_i} w_{ij} x_j^{(k)} - \frac{1}{n} \sum_{j=1}^n \nabla f_j(x_j^{(k)})$$

- Assume all nodes initialize at  $x_i^{(k)} = x^\star$ , we find  $x_i^{(k+1)}$  still remains at  $x^\star$

$$x_i^{(k+1)} = \sum_{j \in \mathcal{N}_i} w_{ij} x^\star - \frac{1}{n} \sum_{j=1}^n \nabla f_j(x^\star) = x^\star - \frac{1}{n} \sum_{j=1}^n \nabla f_j(x^\star) = x^\star$$



# Dynamic average consensus

---

- How to get the dynamic global average  $\frac{1}{n} \sum_{j=1}^n \nabla f_j(x_j^{(k)})$  in a decentralized manner ?
- Recall the dynamic average consensus algorithm

$$x^{(k+1)} = Wx^{(k)} + z^{(k+1)} - z^{(k)} \quad \text{where} \quad x^{(0)} = z^{(0)}$$

- When the dynamic  $z^{(k)}$  converges to stationary points or oscillates slowly, we have

$$x_i^{(k)} \rightarrow \frac{1}{n} \sum_{j=1}^n z_j^{(k)} \quad \text{as} \quad k \rightarrow \infty, \quad \forall i \in [n]$$

# Gradient tracking

---



- Inspired by dynamic average consensus, we can track the globally averaged gradient as follows

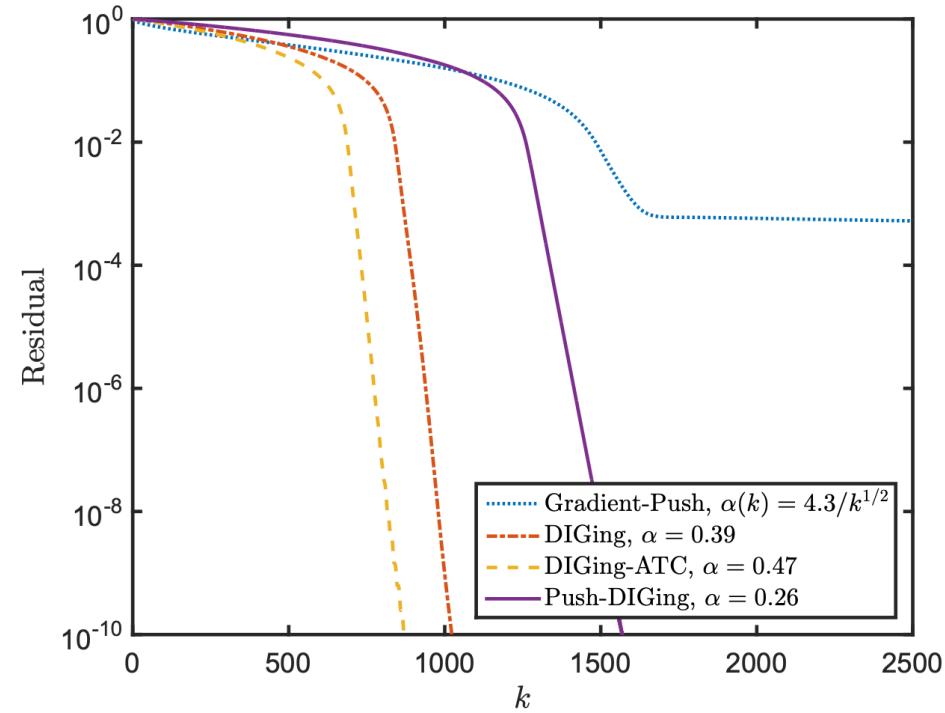
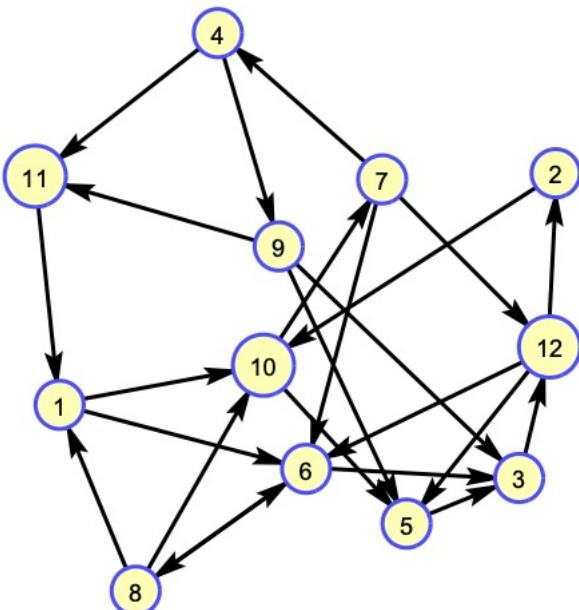
$$\mathbf{y}^{(k+1)} = W\mathbf{y}^{(k)} + \nabla\mathbf{F}(\mathbf{x}^{(k+1)}) - \nabla\mathbf{F}(\mathbf{x}^{(k)}) \quad \text{with} \quad \mathbf{y}^{(0)} = \nabla\mathbf{F}(\mathbf{x}^{(0)})$$

- When  $\nabla\mathbf{F}(\mathbf{x}^{(k+1)})$  converges to stationary points, we have  $y_i^{(k)} \rightarrow \frac{1}{n} \sum_{i=1}^n \nabla f_j(x_j^{(k)})$
- Gradient tracking [XZSX15 ,LS16, NOS17]

$$\mathbf{x}^{(k+1)} = W\mathbf{x}^{(k)} - \gamma\mathbf{y}^{(k)} \tag{DSGD}$$

$$\mathbf{y}^{(k+1)} = W\mathbf{y}^{(k)} + \nabla\mathbf{F}(\mathbf{x}^{(k+1)}) - \nabla\mathbf{F}(\mathbf{x}^{(k)}) \tag{Gradient tracking}$$

# Gradient tracking: simulations



# Gradient tracking: adapt-then-combine

---

- Adapt-then-combine gradient tracking has slightly better performance

(Semi-ATC)

$$\begin{aligned}\mathbf{x}^{(k+1)} &= W(\mathbf{x}^{(k)} - \gamma \mathbf{y}^{(k)}) \\ \mathbf{y}^{(k+1)} &= W\mathbf{y}^{(k)} + \nabla \mathbf{F}(\mathbf{x}^{(k+1)}) - \nabla \mathbf{F}(\mathbf{x}^{(k)})\end{aligned}$$

(Fully-ATC)

$$\begin{aligned}\mathbf{x}^{(k+1)} &= W(\mathbf{x}^{(k)} - \gamma \mathbf{y}^{(k)}) \\ \mathbf{y}^{(k+1)} &= W(\mathbf{y}^{(k)} + \nabla \mathbf{F}(\mathbf{x}^{(k+1)}) - \nabla \mathbf{F}(\mathbf{x}^{(k)}))\end{aligned}$$