Newton-Raphson Consensus for Distributed Convex Optimization

Damiano Varagnolo, Filippo Zanella, Angelo Cenedese, Gianluigi Pillonetto, Luca Schenato

***Abstract*—We address the problem of distributed unconstrained convex optimization under separability assumptions, i.e., the framework where a network of agents, each endowed with a local private multidimensional convex cost and subject to communication constraints, wants to collaborate to compute the minimizer of the sum of the local costs. We propose a design methodology that combines average consensus algorithms and separation of time-scales ideas. This strategy is proved, under suitable hypotheses, to be globally convergent to the true minimizer. Intuitively, the procedure lets the agents distributedly compute and sequentially update an approximated Newton-Raphson direction by means of suitable average consensus ratios. We show with numerical simulations that the speed of convergence of this strategy is comparable with alternative optimization strategies such as the Alternating Direction Method of Multipliers. Finally, we propose some alternative strategies which trade-off communication and computational requirements with convergence speed.**

***Index Terms*—Distributed optimization, unconstrained convex optimization, consensus, multi-agent systems, Newton-Raphson methods, smooth functions.**

I. Introduction

Optimization is a pervasive concept underlying many aspects of modern life [3], [4], [5], and it also includes the management of distributed systems, i.e., artifacts composed by a multitude of interacting entities often referred to as "agents". Examples are transportation systems, where the agents are both the vehicles and the traffic management devices (traffic lights), and smart electrical grids, where the agents are the energy producers-consumers and the power transformers-transporters.

Here we consider the problem of distributed optimization, i.e., the class of algorithms suitable for networked systems and characterized by the absence of a centralized coordination unit [6], [7], [8]. Distributed optimization tools have received an increasing attention over the last years, concurrently with the research on networked control systems. Motivations comprise the fact that the former methods let the networks self-organize and adapt to surrounding and changing environments, and that they are necessary to manage extremely complex systems in an autonomous way with only limited human intervention. In particular we focus on unconstrained convex optimization, although there is a rich literature also on distributed constrained optimization such as Linear Programming [9].

Literature review

The literature on distributed unconstrained convex optimization is extremely vast and a first taxonomy can be based whether the strategy uses or not the Lagrangian framework, see, e.g., [5, Chap. 5].

D. Varagnolo is with the School of Electrical Engineering, KTH Royal Institute of Technology, Osquldas väg 10, Stockholm, Sweden. Email: damiano@kth.se. F. Zanella, A. Cenedese, G. Pillonetto and L. Schenato are with the Department of Information Engineering, Università di Padova, Via Gradenigo 6/a, Padova, Italy. Emails: {fzanella | angelo.cenedese | giapi | schenato }@dei.unipd.it.

The research leading to these results has received funding from the European Union Seventh Framework Programme [FP7/2007-2013] under grant agreement n°257462 HYCON2 Network of excellence, from the Knuth and Alice Wallenberg Foundation and the Swedish Research Council. This paper is an extended and revised version of [1], [2].

Among the distributed methods exploiting Lagrangian formalism, the most widely known algorithm is Alternating Direction Method of Multipliers (ADMM) [10], whose roots can be traced back to [11]. Its efficacy in several practical scenarios is undoubted, see, e.g., [12] and references therein. A notable size of the dedicated literature focuses on the analysis of its convergence performance and on the tuning of its parameters for optimal convergence speed, see, e.g., [13] for Least Squares (LS) estimation scenarios or [14] for linearly constrained convex programs. Even if proved to be an effective algorithm, ADMM suffers from requiring synchronous communication protocols, although some recent attempts for asynchronous and distributed implementations have appeared [15], [16], [17].

On the other hand, among the distributed methods not exploiting Lagrangian formalisms, the most popular ones are the Distributed Subgradient Methods (DSMs) [18]. Here the optimization of non-smooth cost functions is performed by means of subgradient based descent/ascent directions. These methods arise in both primal and dual formulations, since sometimes it is better to perform dual optimization. Subgradient methods have been exploited for several practical purposes, e.g., to optimally allocate resources in Wireless Sensor Networks (WSNs) [19], to maximize the convergence speeds of gossip algorithms [20], to manage optimality criteria defined in terms of ergodic limits [21]. Several works focus on the analysis of the convergence properties of the DSM basic algorithm [22], [23], [24] (see [25] for a unified view of many convergence results). We can also find analyses for several extensions of the original idea, e.g., directions that are computed combining information from other agents [26], [27] and stochastic errors in the evaluation of the subgradients [28]. Explicit characterizations can also show trade-offs between desired accuracy and number of iterations [29].

These methods have the advantage of being easily distributed, to have limited computational requirements and to be inherently asynchronous as shown in [30], [31], [32]. However they suffer from low convergence rate since they require the update steps to decrease to zero as $1/t$ (being t the time) therefore as a consequence the rate of convergence is sub-exponential. In fact, one of the current trends is to design strategies that improve the convergence rate of DSMs. For example, a way is to accelerate the convergence of subgradient methods by means of multi-step approaches, exploiting the history of the past iterations to compute the future ones [33]. Another is to use Newton-like methods, when additional smoothness assumptions can be used. These techniques are based on estimating the Newton direction starting from the Laplacian of the communication graph. More specifically, distributed Newton techniques have been proposed in dual ascent scenarios [34], [35], [36]. Since the Laplacian cannot be computed exactly, the convergence rates of these schemes rely on the analysis of inexact Newton methods [37]. These Newton methods are shown to have super-linear convergence under specific assumptions, but can be applied only to specific optimization problems such as network flow problems.

Recently, several alternative approaches to ADMM and DSM have appeared. For example, in [38], [39] the authors construct contraction mappings by means of cyclic projections of the estimate of the optimum onto the constraints. A similar idea based on contraction maps is used in F-Lipschitz methods [40] but it requires additional assumptions on the cost functions. Other methods are the control-based approach [41] which exploits distributed consensus, and the distributed randomized Kaczmarz method [42] for quadratic cost functions.

Statement of contributions

Here we propose a distributed Newton-Raphson optimization procedure, named Newton-Raphson Consensus (NRC), for the exact minimization of smooth multidimensional convex separable problems, where the global function is a sum of private local costs. With respect to the categorization proposed before, the strategy exploits neither Lagrangian formalisms nor Laplacian estimation steps. More specifically, it is based on average consensus techniques [43] and on the principle of separation of time-scales [44, Chap. 11]. The main idea is that agents compute and keep updated, by means of average consensus protocols, an approximated Newton-Raphson direction that is built from suitable Taylor expansions of the local costs. Simultaneously, agents move their local guesses towards the Newton-Raphson direction. It is proved that, if the rate of change of the local update steps is sufficiently slow to allow the consensus algorithm to converge, then the NRC algorithm exponentially converges to global minimizer.

The proposed algorithm has several advantages w.r.t. the aforementioned literature. The first advantage is that it is as easy to implement as the DSM but, if opportunely tuned, it generally shows faster convergence rates. The second is that it inherits the properties of average consensus algorithms, i.e., it can be asynchronous, and it can be adapted for the time-varying network topologies as in [45] or directed graphs as in [46]. Finally, we also show via numerical simulations based on relevant applications in machine learning and real data that our Newton-Raphson consensus approach can have comparable or better convergence rates of standard ADMM algorithms.

Structure of the paper

The paper is organized as follows: Section II collects the notation used through the whole paper. Section III formulates the problem and reports some preliminary results. Section IV proposes the main optimization algorithm in a scalar scenario based on sensible intuitions and provides convergence and robustness results. Section V generalizes this algorithm to multidimensional domains and offers some strategies to reduce communication and computational complexity. Section VI compares the performance of the proposed algorithm with several distributed optimization strategies available in the literature via numerical simulations. Finally, Section VII collects some final observations and suggests future research directions.

II. Notation

We model the communication network as a graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ whose vertexes $\mathcal{N} := \{1, 2, \ldots, N\}$ represent the agents and whose edges $(i, j) \in \mathcal{E}$ represent the available communication links. We assume that the graph is undirected and connected, and that the matrix $P \in \mathbb{R}^{N\times N}$ is stochastic, i.e., its elements are non-negative, it is s.t. $P\mathbb{1} = \mathbb{1}$ where $\mathbb{1} := [1\ 1\ \cdots\ 1]^T \in \mathbb{R}^N$, and it is consistent with the graph $\mathcal{G}$, in the sense that each entry p_{ij} of P is $p_{ij} > 0$ only if $(i, j) \in \mathcal{E}$. We recall that if P is also symmetric and includes all edges, i.e., $p_{ij} > 0$ if and only if $(i, j) \in \mathcal{E}$, the previous assumptions ensure that $\lim_{k\to\infty} P^k = \frac{1}{N}\mathbb{1}\mathbb{1}^T$. Such P's are also often referred to as *average consensus matrices*. We will indicate with $\rho(P) = \max_{i,\lambda_i\neq 1} |\lambda_i(P)|$ the spectral radius of P, with $\sigma(P) = 1 - \rho(P)$ its spectral gap.

We use plain italic lower case fonts to indicate scalar quantities or functions whose range is a scalar (e.g., x, y, z), bold italic lower case fonts to indicate vectorial quantities or functions whose range is vectorial (e.g., $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{z}$), plain italic capital letters to denote matrices or outcomes of Kronecker products (e.g., X, Y, Z). We use Kronecker products also to indicate component-wise consensus steps. That is, if

$$A_i = \begin{bmatrix} a_{11}^{(i)} & \cdots & a_{1L}^{(i)} \\ \vdots & & \vdots \\ a_{M1}^{(i)} & \cdots & a_{ML}^{(i)} \end{bmatrix} \qquad i = 1, \ldots, N$$

is a generic $M \times L$ matrix associated to agent i, $i = 1, \ldots, N$, and if these agents want to distributedly compute $\frac{1}{N}\sum_{i=1}^{N} A_i$ by means of the communication matrix P, then to indicate the whole set of the single component-wise steps

$$\begin{bmatrix} a_{ml}^{(1)}(k+1) \\ \vdots \\ a_{ml}^{(N)}(k+1) \end{bmatrix} = P \begin{bmatrix} a_{ml}^{(1)}(k) \\ \vdots \\ a_{ml}^{(N)}(k) \end{bmatrix} \qquad \begin{matrix} m = 1, \ldots, M \\ l = 1, \ldots, L \end{matrix} \tag{1}$$

we use the equivalent matrix notation

$$\begin{bmatrix} A_1(k+1) \\ \vdots \\ A_N(k+1) \end{bmatrix} = (P \otimes I_M) \begin{bmatrix} A_1(k) \\ \vdots \\ A_N(k) \end{bmatrix} \tag{2}$$

where I_M is the identity in $\mathbb{R}^{M\times M}$ and $\otimes$ is the Kronecker product. Notice that this notation is suited also for vectorial quantities, i.e., it holds also if $A_i \in \mathbb{R}^M$. We also use fraction bars to indicate Hadamard divisions, e.g., if $\boldsymbol{a} = [a_1, \ldots, a_N]^T$ and $\boldsymbol{b} = [b_1, \ldots, b_N]^T$ then $\dfrac{\boldsymbol{a}}{\boldsymbol{b}} = \left[\dfrac{a_1}{b_1} \cdots \dfrac{a_N}{b_N}\right]^T$. Also, M indicates the dimensionality of the domain, k a discrete time index, t a continuous time index.

Finally, if f is a scalar function, we denote differentiation with $f' := \dfrac{df}{dx}$ and $f'' := \dfrac{d^2 f}{dx^2}$ when the domain is scalar, and with ∇ operators when it is not.

In the Appendix, all the additional notation is collected to provide a quick reference list.

III. Problem formulation and preliminary results

We start dealing with the scalar case, and assume that the N agents of the network are endowed with cost functions $f_i : \mathbb{R} \mapsto \mathbb{R}$ so that

$$\overline{f} : \mathbb{R} \mapsto \mathbb{R}, \qquad \overline{f}(x) := \frac{1}{N}\sum_{i=1}^{N} f_i(x) \tag{3}$$

is a well-defined global cost. We assume that the aim of the agents is to cooperate and distributedly compute the minimizer of $\overline{f}$, namely

$$x^* := \arg\min_x \overline{f}(x). \tag{4}$$

We now enforce the following simplificative assumptions, see, e.g., [27], [47], stated in general for the multidimensional case and valid throughout the rest of the paper:

Assumption 1 (Convexity) $\overline{f}$ in (3) is of class $\mathcal{C}^2$, coercive, strictly convex and with strictly positive second derivatives, i.e., $\overline{f}''(x) := \dfrac{d^2\overline{f}(x)}{dx^2} > 0$, $\forall x \in \mathbb{R}$.

Assumption 1 ensures x^* in (4) to exists and be unique. The positive second derivative is moreover a mild sufficient condition to guarantee that the minimum x^* defined in (4) will be exponentially stable under the continuous Newton-Raphson dynamics described in the following Theorem 2. Notice that in principle just the average function $\overline{f}$ needs to have specific properties, and thus no conditions for the single f_i's are required: in fact they might even be non convex.

The following theorem is a preliminary result on the applicability of Newton-Raphson (NR) optimization procedures[1]. It will be used to prove the convergence properties of the algorithms proposed hereafter. Moreover, along with the $\mathcal{C}^2$ requirements in Assumption 1, the theorem will allow us to apply standard singular perturbation analysis techniques, see, e.g., [44, Chap. 11] [50].

Theorem 2 For every $r > \overline{f}(x^*)$, let $D_r := \left\{x \in \mathbb{R} \mid \overline{f}(x) \leq r\right\}$. Let moreover

$$\dot{x}(t) = -\frac{\overline{f}'\big(x(t)\big)}{\overline{f}''\big(x(t)\big)} =: \psi\big(x(t)-x^*\big), \quad x(0) \in D_r \tag{5}$$

describe a continuous-time Newton-Raphson algorithm with $\overline{f}$ satisfying Assumption 1. Then x^* is an exponentially stable equilibrium, i.e., $|x(t) - x^*| \leq c e^{-\gamma t}|x(0) - x^*|$, $\forall$ t's and $\forall x(0) \in D_r$, for suitable positive constants c and γ possibly depending on r.

Proof We proceed showing that $\overline{f}$ is itself a suitable Lyapunov function for (5). Then we exploit stability theorems offered in [44]. We now show that $\overline{f}$ is suitable Lyapunov function for (5). Then, since $\overline{f}$ is coercive, convex and smooth, the set D_r is closed, convex and compact. Moreover $x^* \in D_r$. Let then $a_1 := \min_{x \in D_r} \overline{f}''(x)$ and $a_2 := \max_{x \in D_r} \overline{f}''(x)$, whose existence is assured being D_r closed and compact. Moreover $0 < a_1 \leq a_2$, since $\overline{f}''(x) > 0$ by hypothesis.

Consider then a generic $x \in D_r$, and the Taylor expansion of $\overline{f}$ around x^* with remainder in Lagrange form, i.e.,

$$\overline{f}(x) = \overline{f}(x^*) + \overline{f}'(x^*)(x - x^*) + \frac{\overline{f}''(\widetilde{x})}{2}(x - x^*)^2 \tag{6}$$

for a suitable $\widetilde{x}$ between x and x^* (thus $\widetilde{x} \in D_r$ by convexity). Since $\overline{f}'(x^*) = 0$, we can transform (6) into $\overline{f}(x) - \overline{f}(x^*) = \frac{\overline{f}''(\widetilde{x})}{2}(x - x^*)^2$, i.e.,

$$\frac{a_1}{2}(x - x^*)^2 \leq \overline{f}(x) - \overline{f}(x^*) \leq \frac{a_2}{2}(x - x^*)^2, \qquad \forall x \in D_r. \tag{7}$$

Moreover, differentiating (6) we obtain $\overline{f}'(x) = \overline{f}'(x^*) + \overline{f}''(\widetilde{x})(x - x^*)$, which implies

$$a_1 |x - x^*| \leq \left|\overline{f}'(x)\right| \leq a_2 |x - x^*|, \qquad \forall x \in D_r. \tag{8}$$

Consider then system (5). Exploiting (8) it follows that, $\forall x(t) \in D_r - \{x^*\}$,

$$\dot{\overline{f}}\big(x(t)\big) = \overline{f}'\big(x(t)\big)\dot{x}(t) = -\frac{\Big(\overline{f}'\big(x(t)\big)\Big)^2}{\overline{f}''\big(x(t)\big)} \leq -\frac{a_1^2}{a_2}\big(x(t) - x^*\big)^2 < 0. \tag{9}$$

Consider then Theorem 4.10 in [44, p. 154]. Here (7) corresponds to (4.25), (9) corresponds to (4.26), and all the other hypotheses are satisfied. Thus $\overline{f}$ is a valid Lyapunov function and x^* is exponentially stable for all $x(0) \in D_r$. ■

[1]Other asymptotic properties of continuous time NR methods can be found, e.g., in [48], [49].

We notice that Theorem 2 states that x^* is practically globally stable. Thus one can start from any point and have an exponential convergence, although a convergence rate that is uniform for all initial conditions might not exist. Nonetheless we can notice that, locally and around the optimum, the rate of convergence of the Newton-Raphson dynamics is $\gamma = 1$ independently of the convex function $\overline{f}$. In fact, if we linearize ψ around 0 (i.e., the dynamics of (5) around x^*) we obtain

$$\begin{aligned}\psi(x) &= \psi(0) + \psi'(0)x + o(x)\\ &= -\frac{\overline{f}'(x^*)}{\overline{f}''(x^*)} - \frac{\overline{f}''(x^*)\overline{f}''(x^*) - \overline{f}'(x^*)\overline{f}'''(x^*)}{\Big(\overline{f}''(x^*)\Big)^2}x + o(x)\\ &= -x + o(x)\end{aligned}$$

since $\overline{f}'(x^*) = 0$ and $\overline{f}''(x^*) \neq 0$.

IV. Newton-Raphson Consensus – the scalar case

For a better understanding of the algorithm we are going to propose, we add some additional assumptions and we later generalize these ideas into the general framework.

We start analyzing the following simplified scenario: the local costs are quadratic and scalar, i.e., $f_i(x) = \frac{1}{2}a_i(x - b_i)^2$, with $a_i > 0$ and $x \in \mathbb{R}$. It is known that, in this case, $x^* = \arg\min_{x \in \mathbb{R}} \overline{f}(x)$ can be computed using two average consensus algorithms in parallel, see, e.g., [51], [52]. In fact

$$x^* = \frac{\sum_{i=1}^N a_i b_i}{\sum_{i=1}^N a_i} = \frac{\frac{1}{N}\sum_{i=1}^N a_i b_i}{\frac{1}{N}\sum_{i=1}^N a_i}, \tag{10}$$

i.e., x^* corresponds to the ratio of two arithmetic means. Thus, *under quadratic costs assumptions*, if each agent defines the local variables $y_i(0) := a_i b_i$ and $z_i(0) := a_i$ and updates them cycling the steps

$$\begin{aligned}\boldsymbol{x}(k+1) &= \frac{\boldsymbol{y}(k)}{\boldsymbol{z}(k)}\\ \boldsymbol{y}(k+1) &= P\boldsymbol{y}(k)\\ \boldsymbol{z}(k+1) &= P\boldsymbol{z}(k),\end{aligned} \tag{11}$$

it follows that, given the fact that P is an average consensus matrix, $\lim_{k\to\infty} \boldsymbol{x}(k) = x^*\mathbb{1}$. Since $x_i(k) = y_i(k)/z_i(k) \to x^*$ for all i's, the $x_i(k)$'s computed through (11) can be thought as the local estimates at time k of the global minimizer x^*.

We can now generalize (11) to the case where the local cost functions f_i are *not* quadratic. Consider then the definitions

$$g_i(x) := f_i''(x)x - f_i'(x), \qquad h_i(x) := f_i''(x),$$

and observe that, for all x, in the quadratic case it holds that $a_i b_i = f_i''(x)x - f_i'(x) =: g_i(x)$ and that $a_i = f_i''(x) =: h_i(x)$. As a consequence, we could let each agent choose an $x_i(0)$ for all i, then set $y_i(0) = f_i''\big(x_i(0)\big)x_i(0) - f_i'\big(x_i(0)\big)$ and $z_i(0) = f_i''\big(x_i(0)\big)$, apply (11) up to convergence and thus compute

$$\widehat{x}^* = \frac{\frac{1}{N}\sum_{i=1}^N \Big(f_i''\big(x_i(0)\big)x_i(0) - f_i'\big(x_i(0)\big)\Big)}{\frac{1}{N}\sum_{i=1}^N f_i''\big(x_i(0)\big)} = \frac{\frac{1}{N}\sum_{i=1}^N g_i\big(x_i(0)\big)}{\frac{1}{N}\sum_{i=1}^N h_i\big(x_i(0)\big)}. \tag{12}$$

Because of the intuitions given before, we expect $\widehat{x}^*$ to be a sensible estimation for the true minimizer x^*. However, $\widehat{x}^*$ depends on the initial conditions $x_i(0)$ and, in general, if the f_i's are not quadratic then $\widehat{x}^* \neq x^*$. Therefore, (11) cannot be applied directly. Nonetheless we notice that if all the $x_i(0)$'s are equal, i.e., $x_i(0) = x$, $\forall i$, then

$$\widehat{x}^* = x - \frac{\overline{f}'(x)}{\overline{f}''(x)}$$

which is a standard NR update step. Thus, if all the agents agree on the $x_i(0)$'s and the consensus step (11) is given enough time to converge, then $\widehat{x}^*$ provides the right descent direction. If instead the agents do not agree on the $x_i(0)$'s, then $\widehat{x}^*$ provides just an approximation of the right descent direction.

To design the main algorithm we then observe the following:

- (11) shall be modified so that it computes meaningful Newton directions even if the $x_i(k)$'s change over time. Since time-varying $x_i(k)$'s imply time-varying $g_i\big(x_i(k)\big)$'s and $h_i\big(x_i(k)\big)$'s, the $y_i(k)$'s and $z_i(k)$'s must track the changing averages $\frac{1}{N}\sum_{i=1}^{N} g_i\big(x_i(k)\big)$ and $\frac{1}{N}\sum_{i=1}^{N} h_i\big(x_i(k)\big)$;
- the computation of the averages of the various $g_i\big(x_i(k)\big)$'s and $h_i\big(x_i(k)\big)$'s must have a convergence rate that is sufficiently faster than the rate of change of the different $x_i(k)$'s.

These ideas are captured in the following Algorithm 1, where the vectorial notation also for the functions $\boldsymbol{g}\big(\boldsymbol{x}(k)\big)$ and $\boldsymbol{h}\big(\boldsymbol{x}(k)\big)$ is introduced.

Algorithm 1 Newton-Raphson Consensus (NRC) – scalar case

```
   (storage allocation and constraints on the parameters)
1: $\boldsymbol{x}(k), \boldsymbol{y}(k), \boldsymbol{z}(k) \in \mathbb{R}^N$ for all $k$. $\varepsilon \in (0,1)$
   (initialization)
2: $\boldsymbol{x}(0) = \boldsymbol{0}$. $\boldsymbol{y}(0) = \boldsymbol{g}\big(\boldsymbol{x}(-1)\big) = \boldsymbol{0}$. $\boldsymbol{z}(0) = \boldsymbol{h}\big(\boldsymbol{x}(-1)\big) = \mathbb{1}$
   (main algorithm)
3: for $k = 1, 2, \ldots$ do
4:     $\boldsymbol{x}(k) = (1-\varepsilon)\boldsymbol{x}(k-1) + \varepsilon \frac{\boldsymbol{y}(k-1)}{\boldsymbol{z}(k-1)}$
5:     $\boldsymbol{y}(k) = P\Big(\boldsymbol{y}(k-1) + \boldsymbol{g}\big(\boldsymbol{x}(k-1)\big) - \boldsymbol{g}\big(\boldsymbol{x}(k-2)\big)\Big)$
6:     $\boldsymbol{z}(k) = P\Big(\boldsymbol{z}(k-1) + \boldsymbol{h}\big(\boldsymbol{x}(k-1)\big) - \boldsymbol{h}\big(\boldsymbol{x}(k-2)\big)\Big)$
7: end for
```

The following remarks highlight the peculiar structure of Algorithm 1:

- line 4 substitutes the local guess update step $x_i(k) = y_i(k-1)/z_i(k-1)$ in (11) with a low-pass filter dominated by the parameter ε. This is necessary because the consensus process on y_i's and z_i's must be faster than the tendency of $x_i(k)$'s to spread apart while the various y_i's and z_i's are not close. If this spreading mechanism is not dominated by the consensus on the y_i's and z_i's, the algorithm could eventually diverge. In other words, line 4 softens possible too aggressive updates of the local estimates. It is also similar to what is usually done in NR approaches where only a small step is taken towards the newly estimated global minimum;
- the initialization in line 2 is *critical* for convergence to the global minimizer. However robustness analysis on possible numerical errors in the initial conditions or quantization noise is discussed below;
- in lines 5-6 the various agents modify their local y_i and z_i (and thus take into account the effects induced by changing $x_i(k)$'s) *before* performing the consensus steps on the y_i's and z_i's. Basically, lines 5-6 perform a joint high-pass filter w.r.t. time and a tracking of the changing averages $\frac{1}{N}\sum_{i=1}^{N} g_i(x_i)$ and $\frac{1}{N}\sum_{i=1}^{N} h_i(x_i)$;
- if $\varepsilon = 1$ and the functions f_i are quadratic, then Algorithm 1 reduces to the system (11).

Before providing a formal proof of the convergence properties of Algorithm 1 we give some intuitions on its behavior. The dynamics of Algorithm 1 can be written in state space as follows:

$$\left\{\begin{array}{rcl} \boldsymbol{v}(k) &=& \boldsymbol{g}\big(\boldsymbol{x}(k-1)\big) \\ \boldsymbol{w}(k) &=& \boldsymbol{h}\big(\boldsymbol{x}(k-1)\big) \\ \boldsymbol{y}(k) &=& P\Big[\boldsymbol{y}(k-1) + \boldsymbol{g}\big(\boldsymbol{x}(k-1)\big) - \boldsymbol{v}(k-1)\Big] \\ \boldsymbol{z}(k) &=& P\Big[\boldsymbol{z}(k-1) + \boldsymbol{h}\big(\boldsymbol{x}(k-1)\big) - \boldsymbol{w}(k-1)\Big] \\ \boldsymbol{x}(k) &=& (1-\varepsilon)\boldsymbol{x}(k-1) + \varepsilon\dfrac{\boldsymbol{y}(k-1)}{\boldsymbol{z}(k-1)}, \end{array}\right. \tag{13}$$

which can be interpreted as the forward-Euler discrete-time version of continuous-time system

$$\left\{\begin{array}{rcl} \varepsilon\dot{\boldsymbol{v}}(t) &=& -\boldsymbol{v}(t) + \boldsymbol{g}\big(\boldsymbol{x}(t)\big) \\ \varepsilon\dot{\boldsymbol{w}}(t) &=& -\boldsymbol{w}(t) + \boldsymbol{h}\big(\boldsymbol{x}(t)\big) \\ \varepsilon\dot{\boldsymbol{y}}(t) &=& -K\boldsymbol{y}(t) + (I-K)\Big[\boldsymbol{g}\big(\boldsymbol{x}(t)\big) - \boldsymbol{v}(t)\Big] \\ \varepsilon\dot{\boldsymbol{z}}(t) &=& -K\boldsymbol{z}(t) + (I-K)\Big[\boldsymbol{h}\big(\boldsymbol{x}(t)\big) - \boldsymbol{w}(t)\Big] \\ \dot{\boldsymbol{x}}(t) &=& -\boldsymbol{x}(t) + \dfrac{\boldsymbol{y}(t)}{\boldsymbol{z}(t)} \end{array}\right. \tag{14}$$

where ε is the discretization time interval and $K := I - P$. As a consequence, for sufficiently small ε the dynamic behavior of (13) is approximated by the one of (14), i.e., $x(\varepsilon k) \approx x(k)$, where, with a little abuse of notation, we used the same symbols for the continuous and discrete time variables.

It is now possible to recognize the existence of a two-time scales dynamical system regulated by the parameter ε. Therefore, we can split the dynamics in the two time scales and study them separately for sufficiently small ε.

As for the *fast dynamics*, by construction K is positive semidefinite with kernel spanned by $\mathbb{1}$ and with eigenvalues $0 = \lambda_1 < \mathrm{Re}[\lambda_2] \leq \cdots \leq \mathrm{Re}[\lambda_N] < 2$. I.e., the first four equations of system (14) imply that $\boldsymbol{y}(t) \approx \left(\frac{1}{N}\mathbb{1}^T\boldsymbol{g}\big(\boldsymbol{x}(t)\big)\right)\mathbb{1}$ and $\boldsymbol{z}(t) \approx \left(\frac{1}{N}\mathbb{1}^T\boldsymbol{h}\big(\boldsymbol{x}(t)\big)\right)\mathbb{1}$.

If these equations are inserted into the *slow dynamics*, i.e., into the last equation of system (14), then it follows that $\boldsymbol{x}(t) \approx \overline{x}(t)\mathbb{1}$, where $\overline{x}(t)$ is a scalar quantity that approximately evolves following the continuous-time Newton-Raphson update

$$\dot{\overline{x}}(t) = -\frac{\overline{f}'\big(\overline{x}(t)\big)}{\overline{f}''\big(\overline{x}(t)\big)}\,. \tag{15}$$

Summarizing, for sufficiently small values of ε Algorithm 1 can be described by (14), it is s.t. $x_i(k) \approx \overline{x}(\varepsilon k)$ for all i's, and thus by Theorem 2 it converges to the global optimum x^*.

We now move from intuitions to a formal proof of convergence. We start by considering the robustness of the algorithm in terms of possible different initial conditions $\boldsymbol{x}(0)$.

Theorem 3 Consider Algorithm 1 with arbitrary initial conditions $\boldsymbol{x}(0)$ and let Assumption 1 hold true. For every open ball $B_r^{x^*} := \{\boldsymbol{x} \mid \|\boldsymbol{x} - x^*\mathbb{1}\| < r\}$ there exist two positive constants $\overline{\varepsilon}_r$, c_r such that if $\varepsilon < \overline{\varepsilon}_r$, then there exists $\gamma_\varepsilon > 0$ such that, for all $\boldsymbol{x}(0) \in B_r^{x^*}$, $\|\boldsymbol{x}(k) - x^*\mathbb{1}\| \leq c_r e^{-\gamma_\varepsilon k}\|\boldsymbol{x}(0) - x^*\mathbb{1}\|$ for all k.

Proof Since assumptions for Theorem 2 in [53] are satisfied, we are ensured that if ε is sufficiently small then the discretized system (13) inherits the same stability properties of (14). Therefore we focus on proving that Theorem 3 holds true considering system (14) rather than Algorithm 1.

The proof is based on characterizing system (14) through classical multi-time-scales approaches for standard singular perturbation model analysis [50], [44, Chap. 11]. It is divided in the following steps: a) perform some suitable changes of variables, b) analyze the boundary layer system (fast dynamics), c) analyze the reduced system (slow dynamics).

In the following we will use the additional notation

$$\Pi^{\parallel} := \frac{\mathbb{1}\mathbb{1}^T}{N} \qquad \Pi^{\perp} := I - \frac{\mathbb{1}\mathbb{1}^T}{N} \qquad \overline{x} := \frac{1}{N}\sum_{i=1}^{N} x_i$$

$$\boldsymbol{x}^{\parallel} := \Pi^{\parallel}\boldsymbol{x} = \overline{x}\mathbb{1} \qquad \boldsymbol{x}^{\perp} := \Pi^{\perp}\boldsymbol{x}.$$

Moreover, to ease the proof of the following Theorem 4 we initially consider the more general case where $\boldsymbol{g}\big(\boldsymbol{x}(-1)\big)$, $\boldsymbol{h}\big(\boldsymbol{x}(-1)\big)$, $\boldsymbol{y}(0)$, $\boldsymbol{z}(0)$ are generic elements in $\mathbb{R}^N$. To this aim we use the additional notation

$$\alpha := \frac{1}{N}\mathbb{1}^T\big(\boldsymbol{y}(0) - \boldsymbol{v}(0)\big) \qquad \beta := \frac{1}{N}\mathbb{1}^T\big(\boldsymbol{z}(0) - \boldsymbol{w}(0)\big). \quad (16)$$

a) *changes of variables* (to show that the boundary layer system has a single isolated root): let $\boldsymbol{d}(t) := \boldsymbol{y}(t) - \boldsymbol{v}(t)$, so that $\dot{\boldsymbol{y}}(t) = \dot{\boldsymbol{d}}(t) + \dot{\boldsymbol{v}}(t)$. This implies

$$\varepsilon\left(\dot{\boldsymbol{d}}(t) + \dot{\boldsymbol{v}}(t)\right) = -K\big[\boldsymbol{d}(t) + \boldsymbol{v}(t)\big] + (I-K)\Big[\boldsymbol{g}\big(\boldsymbol{x}(t)\big) - \boldsymbol{v}(t)\Big]$$

and thus, since $\varepsilon\dot{\boldsymbol{v}}(t) = -\boldsymbol{v}(t) + \boldsymbol{g}\big(\boldsymbol{x}(t)\big)$,

$$\varepsilon\dot{\boldsymbol{d}}(t) = -K\Big[\boldsymbol{d}(t) + \boldsymbol{g}\big(\boldsymbol{x}(t)\big)\Big]. \quad (17)$$

We now decompose $\boldsymbol{d}(t)$ into $\boldsymbol{d}(t) = \boldsymbol{d}^{\parallel} + \boldsymbol{d}^{\perp}$, $\boldsymbol{d}^{\parallel} := \Pi^{\parallel}\boldsymbol{d}$, $\boldsymbol{d}^{\perp} := \Pi^{\perp}\boldsymbol{d}$, i.e., into "mean component plus deviations from mean". Since $\Pi^{\parallel}K = \boldsymbol{0}$ and $\Pi^{\perp}K = K\Pi^{\perp} = K$, then (17) can be decomposed as

$$\begin{cases} \varepsilon\dot{\boldsymbol{d}}^{\parallel}(t) = \boldsymbol{0} & (18)\\ \varepsilon\dot{\boldsymbol{d}}^{\perp}(t) = -K\Big[\boldsymbol{d}^{\perp}(t) + \boldsymbol{g}\big(\boldsymbol{x}(t)\big)\Big] & (19)\end{cases}$$

with (18) implying $\boldsymbol{d}^{\parallel}(t) = \boldsymbol{d}^{\parallel}(0) = \Pi^{\parallel}\big(\boldsymbol{y}(0) - \boldsymbol{v}(0)\big) = \alpha\mathbb{1}$, α defined in (16).

The same derivations can be applied to the variables $\boldsymbol{b}(t) := \boldsymbol{z}(t) - \boldsymbol{w}(t)$, $\boldsymbol{z}$ and $\boldsymbol{w}$, so that system (14) becomes

$$\begin{cases} \varepsilon\dot{\boldsymbol{v}}(t) &=& -\boldsymbol{v}(t) + \boldsymbol{g}\left(\boldsymbol{x}(t)\right)\\ \varepsilon\dot{\boldsymbol{w}}(t) &=& -\boldsymbol{w}(t) + \boldsymbol{h}\left(\boldsymbol{x}(t)\right)\\ \varepsilon\dot{\boldsymbol{d}}^{\perp}(t) &=& -K\left[\boldsymbol{d}^{\perp}(t) + \boldsymbol{g}\left(\boldsymbol{x}(t)\right)\right]\\ \varepsilon\dot{\boldsymbol{b}}^{\perp}(t) &=& -K\left[\boldsymbol{b}^{\perp}(t) + \boldsymbol{h}\left(\boldsymbol{x}(t)\right)\right]\\ \dot{\boldsymbol{x}}(t) &=& -\boldsymbol{x}(t) + \dfrac{\boldsymbol{d}^{\perp}(t) + \alpha\mathbb{1} + \boldsymbol{v}(t)}{\boldsymbol{b}^{\perp}(t) + \beta\mathbb{1} + \boldsymbol{w}(t)} \end{cases} \quad (20)$$

Notice that the values for α and β reflect the numerical values of the initial conditions of the variables $\boldsymbol{v}$, $\boldsymbol{w}$, $\boldsymbol{y}$, $\boldsymbol{z}$.

b) *analysis of the boundary layer system*: in this case $\boldsymbol{x}(t)$ is considered to be constant in time, so that $\boldsymbol{g}\big(\boldsymbol{x}(t)\big) = \boldsymbol{g}\left(\boldsymbol{x}\right)$ and $\boldsymbol{h}\big(\boldsymbol{x}(t)\big) = \boldsymbol{h}\left(\boldsymbol{x}\right)$. To analyze the stability properties of system

$$\begin{cases} \varepsilon\dot{\boldsymbol{v}}(t) &=& -\boldsymbol{v}(t) + \boldsymbol{g}\left(\boldsymbol{x}\right)\\ \varepsilon\dot{\boldsymbol{w}}(t) &=& -\boldsymbol{w}(t) + \boldsymbol{h}\left(\boldsymbol{x}\right)\\ \varepsilon\dot{\boldsymbol{d}}^{\perp}(t) &=& -K\big[\boldsymbol{d}^{\perp}(t) + \boldsymbol{g}\left(\boldsymbol{x}\right)\big]\\ \varepsilon\dot{\boldsymbol{b}}^{\perp}(t) &=& -K\big[\boldsymbol{b}^{\perp}(t) + \boldsymbol{h}\left(\boldsymbol{x}\right)\big] \end{cases} \quad (21)$$

we start by applying the changes of variables

$$\widetilde{\boldsymbol{v}}(t) := \boldsymbol{v}(t) - \boldsymbol{g}\left(\boldsymbol{x}\right) \qquad \widetilde{\boldsymbol{d}}^{\perp}(t) := \boldsymbol{d}^{\perp}(t) + \Pi^{\perp}\boldsymbol{g}\left(\boldsymbol{x}\right)$$
$$\widetilde{\boldsymbol{w}}(t) := \boldsymbol{w}(t) - \boldsymbol{h}\left(\boldsymbol{x}\right) \qquad \widetilde{\boldsymbol{b}}^{\perp}(t) := \boldsymbol{b}^{\perp}(t) + \Pi^{\perp}\boldsymbol{h}\left(\boldsymbol{x}\right)$$

induced by the isolated root of (21), and the change of timescales $\tau := \frac{t}{\varepsilon} \Rightarrow \frac{d\tau}{dt} = \frac{1}{\varepsilon}$, induced by the low-pass filtering parameter ε. We thus obtain the equivalent boundary layer system

$$\begin{cases} \dot{\widetilde{\boldsymbol{v}}}(\tau) &=& -\widetilde{\boldsymbol{v}}(\tau)\\ \dot{\widetilde{\boldsymbol{w}}}(\tau) &=& -\widetilde{\boldsymbol{w}}(\tau)\\ \dot{\widetilde{\boldsymbol{d}}}^{\perp}(\tau) &=& -K\widetilde{\boldsymbol{d}}^{\perp}(\tau)\\ \dot{\widetilde{\boldsymbol{b}}}^{\perp}(\tau) &=& -K\widetilde{\boldsymbol{b}}^{\perp}(\tau) \end{cases} \quad (22)$$

where the stability properties are equivalent to the ones of (21).

We now show that (22) is exponentially stable. This is clear for the dynamics of the first two equations, while for the last two we claim that $V\left(\boldsymbol{\zeta}\right) = \frac{1}{2}\left\|\boldsymbol{\zeta}\right\|^2$ is a valid Lyapunov function for both $\widetilde{\boldsymbol{d}}^{\perp}$ and $\widetilde{\boldsymbol{b}}^{\perp}$. For, consider that

$$\dot{V}(\widetilde{\boldsymbol{d}}^{\perp}) = -(\widetilde{\boldsymbol{d}}^{\perp})^T K\widetilde{\boldsymbol{d}}^{\perp} \leq -\lambda_2\|\widetilde{\boldsymbol{d}}^{\perp}\|^2 \leq -\lambda_2 V(\widetilde{\boldsymbol{d}}^{\perp}),$$

where $\lambda_2 > 0$ is the smallest non-zero eigenvalue of the matrix K ($\widetilde{\boldsymbol{d}}^{\perp}$ by construction lives in $\ker\left(K\right)^{\perp}$; similar reasonings hold for $\widetilde{\boldsymbol{b}}^{\perp}$).

Applying this result back to (21), we are ensured that (21) is globally exponentially stable with equilibrium given by

$$\lim_{t\to\infty}\begin{bmatrix}\boldsymbol{v}(t)\\ \boldsymbol{w}(t)\\ \boldsymbol{d}^{\perp}(t)\\ \boldsymbol{b}^{\perp}(t)\end{bmatrix} = \begin{bmatrix}\boldsymbol{g}\left(\boldsymbol{x}\right)\\ \boldsymbol{h}\left(\boldsymbol{x}\right)\\ -\Pi^{\perp}\boldsymbol{g}\left(\boldsymbol{x}\right)\\ -\Pi^{\perp}\boldsymbol{h}\left(\boldsymbol{x}\right)\end{bmatrix} \quad (23)$$

for every initial condition and $\boldsymbol{x}$.

c) *analysis of the reduced system*: substituting (23) into the last equation of system (20) we obtain the reduced system

$$\dot{\boldsymbol{x}}(t) = -\boldsymbol{x}(t) + \frac{\alpha\mathbb{1} - \Pi^{\perp}\boldsymbol{g}\big(\boldsymbol{x}(t)\big) + \boldsymbol{g}\big(\boldsymbol{x}(t)\big)}{\beta\mathbb{1} - \Pi^{\perp}\boldsymbol{h}\big(\boldsymbol{x}(t)\big) + \boldsymbol{h}\big(\boldsymbol{x}(t)\big)}\ .$$

Let now $\overline{g}\big(\boldsymbol{x}(k)\big) := \dfrac{1}{N}\displaystyle\sum_{i=1}^{N} g_i\big(x_i(k)\big)$, $\overline{h}\big(\boldsymbol{x}(k)\big) := \dfrac{1}{N}\displaystyle\sum_{i=1}^{N} h_i\big(x_i(k)\big)$, so that we can exploit the equivalence

$$-\Pi^{\perp}\boldsymbol{g}\big(\boldsymbol{x}(t)\big) + \boldsymbol{g}\big(\boldsymbol{x}(t)\big) = \Pi^{\parallel}\boldsymbol{g}\big(\boldsymbol{x}(t)\big) = \overline{g}\big(\boldsymbol{x}(t)\big)\mathbb{1}$$

in the numerator and a similar equivalence for $\boldsymbol{h}\big(\boldsymbol{x}(t)\big)$ in the denominator. We can thus rewrite the reduced system as

$$\dot{\boldsymbol{x}}(t) = -\boldsymbol{x}(t) + \frac{\alpha + \overline{g}\big(\boldsymbol{x}(t)\big)}{\beta + \overline{h}\big(\boldsymbol{x}(t)\big)}\mathbb{1}\ . \quad (24)$$

This eventually implies that (24) can then be rewritten as

$$\dot{\boldsymbol{x}}(t) = \Psi\big(\boldsymbol{x}(t), \alpha, \beta\big) \quad (25)$$

where Ψ is a smooth function of its arguments.

To address the stability of (25) we then decompose its dynamics along the projections given by $\Pi^{\perp}$ and $\Pi^{\parallel}$, obtaining the continuous time non-linear cascade system

$$\begin{cases} \dot{\boldsymbol{x}}^{\parallel}(t) &=& -\boldsymbol{x}^{\parallel}(t) + \dfrac{\overline{g}\Big(\boldsymbol{x}^{\parallel}(t) + \boldsymbol{x}^{\perp}(t)\Big) + \alpha}{\overline{h}\Big(\boldsymbol{x}^{\parallel}(t) + \boldsymbol{x}^{\perp}(t)\Big) + \beta}\mathbb{1}\\ \dot{\boldsymbol{x}}^{\perp}(t) &=& -\boldsymbol{x}^{\perp}(t) \end{cases} \quad (26)$$

where $\boldsymbol{x}^{\perp}(t)$ is independent on $\boldsymbol{x}^{\parallel}(t)$ and exponentially decaying to zero.

We now notice that, by construction, $\boldsymbol{x}^{\parallel}(t) = \overline{x}(t)\mathbb{1}$, i.e., $\boldsymbol{x}^{\parallel}(t)$ is a vector with identical entries. Therefore the dynamic behavior of the first equation in (26) is summarized by

$$\dot{\overline{x}}(t) = -\overline{x}(t) + \frac{\overline{g}\Big(\boldsymbol{x}^{\parallel}(t) + \boldsymbol{x}^{\perp}(t)\Big) + \alpha}{\overline{h}\Big(\boldsymbol{x}^{\parallel}(t) + \boldsymbol{x}^{\perp}(t)\Big) + \beta}. \tag{27}$$

By Theorem 1 in [54][2], proof of exponential stability of system (26) can be reduced to proof of exponential stability of system (27) with $\boldsymbol{x}^{\perp}(t) = 0$, so that our case reduces to analyze

$$\dot{\overline{x}}(t) = -\overline{x}(t) + \frac{\overline{g}\big(\overline{x}(t)\big) + \alpha}{\overline{h}\big(\overline{x}(t)\big) + \beta} = \psi\big(\overline{x}(t), \alpha, \beta\big). \tag{28}$$

We now consider the case $\alpha = \beta = 0$, which is guaranteed by the initialization of line 2 in Algorithm 1. Given $\alpha = \beta = 0$ and considering the definitions of $\overline{g}(x)$ and $\overline{h}(x)$, (28) reduces to the continuous-time Newton-Raphson method

$$\dot{\overline{x}}(t) = -\frac{\overline{f}'\big(\overline{x}(t)\big)}{\overline{f}''\big(\overline{x}(t)\big)} \tag{29}$$

that, due to Theorem 2, exponentially converges to x^*. Moreover, given $\alpha = \beta = 0$, the strict positivity of $\overline{h}(\cdot)$ and the properness of $\overline{g}(\cdot)$, the trajectories of system (27) are all bounded. Thus the hypotheses of Theorem 1 in [54] are all satisfied and we can claim that (27) exponentially converges to $x^*\mathbb{1}$.

It is immediate now to check that the hypotheses of Theorem 11.4 in [44, p. 456] are satisfied, and this guarantees our claims. ■

Theorem 3 holds only for the specific initial conditions given in line 2 of Algorithm 1. Although these initial conditions can be arbitrarily designed, nonetheless it is important to evaluate the robustness of the algorithm with respect to them, since small numerical errors and quantization noise might lead to some perturbations. The following theorem shows that non-null but sufficiently small initial conditions on the variables $\boldsymbol{v}(0), \boldsymbol{w}(0), \boldsymbol{y}(0), \boldsymbol{z}(0)$ let the solution of the algorithm exponentially converge to a neighborhood of the true optimum:

Theorem 4 Consider Algorithm 1 with arbitrary initial conditions, $\boldsymbol{x}(0)$ $\boldsymbol{v}(0)$, $\boldsymbol{w}(0)$, $\boldsymbol{y}(0)$, $\boldsymbol{z}(0)$. Let Assumption 1 hold true and $\alpha := \dfrac{1}{N}\mathbb{1}^T\big(\boldsymbol{y}(0) - \boldsymbol{v}(0)\big)$, $\beta := \dfrac{1}{N}\mathbb{1}^T\big(\boldsymbol{z}(0) - \boldsymbol{w}(0)\big)$. Then there exist two positive constants $\overline{\alpha}$, $\overline{\beta} \in \mathbb{R}_+$ and a scalar smooth function $\xi(\alpha, \beta)$ with $\xi(0, 0) = x^*$ s.t. if $|\alpha| < \overline{\alpha}$ and $|\beta| < \overline{\beta}$ then Theorem 3 holds true with x^* substituted with $\xi(\alpha, \beta)$.

Proof The proof follows exactly as in Theorem 3 up to equation (28), where ψ is a smooth function of its arguments and $\dot{\overline{x}}(t) = \psi\big(\overline{x}(t), 0, 0\big)$ globally and exponentially converges to x^*.

Given our smoothness assumptions, since $\psi(x^*, 0, 0) = 0$ we can apply the Implicit Function Theorem and be ensured that there must exist, in a neighborhood of $\alpha = \beta = 0$, a smooth function $\xi(\alpha, \beta)$ such that $\xi(0, 0) = x^*$ and $\psi\big(\xi(\alpha, \beta), \alpha, \beta\big) = 0$, i.e., $\xi(\alpha, \beta)$ returns the equilibria of system (28) for sufficiently small values of α, β. Then, by performing the change of variables $\overline{\chi} = \overline{x} - \xi(\alpha, \beta)$ and following the same derivations to prove the stability of slowly varying systems in [44, Section 9.6], it readily follows that the equilibrium points $\xi(\alpha, \beta)$ are exponentially stable in a neighborhood of $\alpha = \beta = 0$. ■

The previous theorem shows that the initialization of the variables $\boldsymbol{v}(0), \boldsymbol{w}(0), \boldsymbol{y}(0), \boldsymbol{z}(0)$ is critical to the convergence of the correct minimum, but it also assures that sufficiently small errors will have no dramatic effects such as instability. Numerical simulations in fact suggest that the algorithm is robust w.r.t. numerical errors and quantization noise.

Before turning to the multidimensional scenario, we notice that Theorem 3 guarantees the existence of a critical value $\overline{\varepsilon}_r$ but does not provide indications on its value. This is a known issue in all the systems dealing with separation of time scales. A standard rule of thumb is then to let the rate of convergence of the fast dynamics be sufficiently faster than the one of the slow dynamics, typically 2-10 times faster. In our algorithm the fast dynamics inherits the rate of convergence of the consensus matrix P, given by its spectral gap $\sigma(P)$, i.e., its spectral radius $\rho(P) = 1 - \sigma(P)$. The rate of convergence of the slow dynamics is instead governed by (15), which is nonlinear and therefore possibly depending on the initial conditions. However, close the equilibrium point the dynamic behavior is approximately given by $\dot{\overline{x}}(t) \approx -\big(\overline{x}(t) - x^*\big)$, thus, since $x_i(k) \approx \overline{x}(\varepsilon k)$, then the convergence rate of the algorithm approximately given by $1 - \varepsilon$.

Thus we aim to let $1 - \rho(P) \gg 1 - (1 - \varepsilon)$, which provides the rule of thumb

$$\varepsilon \ll \sigma(P)\ . \tag{30}$$

which is suitable for generic cost functions. We then notice that, although the spectral gap $\sigma(P)$ might not be known in advance, it is possible to distributedly estimate it, see, e.g., [55]. However, such rule of thumb might be very conservative. In fact, as noticed after Algorithm 1, if all the f_i's are quadratic then the procedure reduces to system 11. Thus:

Theorem 5 Consider Algorithm 1 with arbitrary initial conditions $\boldsymbol{x}(0)$, quadratic cost functions, i.e., $f_i = \frac{1}{2}a_i(x - b_i)^2, a_i > 0$, and $\varepsilon = 1$. Then $\|\boldsymbol{x}(k) - x^*\mathbb{1}\| \leq c\,(\rho(P))^k$ for all k and for an opportune positive c.

Proof Let $y^* := \frac{1}{N}\sum_i a_i b_i$ and $z^* := \frac{1}{N}\sum_i a_i$, so that $x^* = \dfrac{y^*}{z^*}$. Since $\mathbf{y}(k+1) = P\mathbf{y}(k)$ and $\mathbf{z}(k+1) = P\mathbf{z}(k)$, given the assumptions on P, there exist positive c_y, c_z independent on $\boldsymbol{x}(0)$ s.t. $|y_i(k) - y^*| \leq c_y\,(\rho(P))^k$ and $|z_i(k) - z^*| \leq c_z\,(\rho(P))^k$. The claim thus follows considering that $x_i(k) = \frac{y_i(k)}{z_i(k)}$ and that, since the elements of P are non negative, all the $z_i(k)$ are strictly positive for all $k \geq 0$. ■

Thus, if the cost functions are close to be quadratic then the overall rate of convergence is limited by the rate of convergence of the embedded consensus algorithm. Moreover, the values of ε that still guarantee convergence can be much larger than those dictated by the rule of thumb (30).

V. Newton-Raphson Consensus – the multidimensional case

In the previous sections we derived the algorithm for the scalar case considering that, for scalar quadratic local costs, the optimum is given by (10). We could derive the algorithm for the multidimensional case using exactly the same intuitions: in fact considering multidimensional quadratic local costs $f_i(\boldsymbol{x}) = \dfrac{1}{2}(\boldsymbol{x} - \boldsymbol{b}_i)^T A_i (\boldsymbol{x} - \boldsymbol{b}_i)$, with $\boldsymbol{x} := [x_1 \cdots x_M]^T$ and $\boldsymbol{b}_i, A_i$ of suitable dimensions, it follows immediately that $\boldsymbol{x}^* = \left(\dfrac{1}{N}\displaystyle\sum_{i=1}^N A_i\right)^{-1}\left(\dfrac{1}{N}\displaystyle\sum_{i=1}^N A_i \boldsymbol{b}_i\right)$.

A sensible extension of the scalar algorithm to a multidimensional scenario is to replace $f_i'(x_i)$ with the gradient $\nabla f_i(\boldsymbol{x}_i)$ and the

[2]Formally, Theorem 1 in [54] considers just simple stability. Nonetheless it is immediate to check that its proof is s.t. if the subsystems are exponentially stable then the overall system is again exponentially stable.

$f_i''(x_i)$ with the full Hessian $\nabla^2 f_i(\boldsymbol{x}_i)$. However, this is not the only possible choice, and indeed, by appropriately defining functions $\boldsymbol{g}_i(\boldsymbol{x}_i)$ and $H_i(\boldsymbol{x}_i)$, which play the role of $g_i(\boldsymbol{x}_i)$ and $h_i(\boldsymbol{x}_i)$ of the scalar case, one can obtain different procedures with different convergence properties and different computational/communication requirements. The following are (some) plausible choices for H_i:

$$H_i\big(\boldsymbol{x}_i(k)\big) = \nabla^2 f_i\big(\boldsymbol{x}_i(k)\big) \in \mathbb{R}^{M\times M} \tag{31}$$

$$\begin{aligned} H_i\big(\boldsymbol{x}_i(k)\big) &= \operatorname{diag}\Big[\nabla^2 f_i\big(\boldsymbol{x}_i(k)\big)\Big] \\ &= \begin{bmatrix} \left.\dfrac{\partial^2 f_i}{\partial x_1^2}\right|_{\boldsymbol{x}_i(k)} & & 0 \\ & \ddots & \\ 0 & & \left.\dfrac{\partial^2 f_i}{\partial x_M^2}\right|_{\boldsymbol{x}_i(k)} \end{bmatrix} \in \mathbb{R}^{M\times M} \end{aligned} \tag{32}$$

$$H_i\big(\boldsymbol{x}_i(k)\big) = I_M \in \mathbb{R}^{M\times M}. \tag{33}$$

Algorithm 2 Multidimensional case – Newton-Raphson, Jacobi and Gradient Descent Consensus

```
    (storage allocation and constraints on the parameters)
 1: $\boldsymbol{x}_i(k), \boldsymbol{y}_i(k) \in \mathbb{R}^M$, $Z_i(k) \in \mathbb{R}^{M\times M}$ for all $i$'s. $\varepsilon \in (0,1)$. $H_i$ defined in (31) or (32) or (33)
    (initialization)
 2: $\boldsymbol{x}_i(0) = \mathbf{0}$. $\boldsymbol{y}_i(0) = \boldsymbol{g}_i\big(\boldsymbol{x}_i(-1)\big) = \mathbf{0}$. $Z_i(0) = H_i\big(\boldsymbol{x}_i(-1)\big) = I$, for all $i$'s
    (main algorithm)
 3: for $k = 1, 2, \ldots$ do
 4:     for $i = 1, \ldots, N$ do
 5:         $\boldsymbol{x}_i(k) = (1-\varepsilon)\boldsymbol{x}_i(k-1) + \varepsilon\big(Z_i(k-1)\big)^{-1}\boldsymbol{y}_i(k-1)$
 6:         $\widetilde{\boldsymbol{y}}_i(k) = \boldsymbol{y}_i(k-1) + \boldsymbol{g}_i\big(\boldsymbol{x}_i(k-1)\big) - \boldsymbol{g}_i\big(\boldsymbol{x}_i(k-2)\big)$
 7:         $\widetilde{Z}_i(k) = Z_i(k-1) + H_i\big(\boldsymbol{x}_i(k-1)\big) - H_i\big(\boldsymbol{x}_i(k-2)\big)$
 8:     end for
 9:     for $i = 1, \ldots, N$ do
10:         $\boldsymbol{y}_i(k) = \sum_{j=1}^N p_{ij}\, \widetilde{\boldsymbol{y}}_j(k)$
11:         $Z_i(k) = \sum_{j=1}^N p_{ij}\, \widetilde{Z}_j(k)$
12:     end for
13: end for
```

The multidimensional version of Algorithm 1 is given by Algorithm 2, where H_i is left undefined and depending on its choice, it leads to a different version of the algorithm. The three proposed choices lead to the following algorithms:

• Equation (31) $\rightarrow$ **Newton-Raphson Consensus (NRC)**: in this case it is possible to rewrite Algorithm 2 as done in Section IV and show that, for sufficiently small ε, $\boldsymbol{x}_i(k) \approx \overline{\boldsymbol{x}}(\varepsilon k)$, where $\overline{\boldsymbol{x}}(t)$ evolves according to the continuous-time Newton-Raphson dynamics

$$\dot{\overline{\boldsymbol{x}}}(t) = -\Big[\nabla^2 \overline{f}\big(\overline{\boldsymbol{x}}(t)\big)\Big]^{-1} \nabla \overline{f}\big(\overline{\boldsymbol{x}}(t)\big) \ ,$$

which, analogously to its scalar version, can be shown to converge to the global optimum $\boldsymbol{x}^*$.

• Equation (32) $\rightarrow$ **Jacobi Consensus (JC)**: choice (31) requires agents to exchange information on $O\left(M^2\right)$ scalars, and this could pose problems under heavy communication bandwidth constraints and large M's. Choice (32) instead reduces the amount of information to be exchanged via the underlying diagonalization process, also called Jacobi approximation[3]. In this case, for sufficiently small ε, $\boldsymbol{x}_i(k) \approx \overline{\boldsymbol{x}}(\varepsilon k)$, where $\overline{\boldsymbol{x}}(t)$ evolves according to the continuous-time dynamics

$$\dot{\overline{\boldsymbol{x}}}(t) = -\Big(\operatorname{diag}\Big[\nabla^2 \overline{f}\big(\overline{\boldsymbol{x}}(t)\big)\Big]\Big)^{-1} \nabla \overline{f}\big(\overline{\boldsymbol{x}}(t)\big) \ ,$$

which can be shown to converge to the global optimum $\boldsymbol{x}^*$ with a convergence rate that in general is slower than the Newton-Raphson when the global cost function is skewed.

• Equation (33) $\rightarrow$ **Gradient Descent Consensus (GDC)**: this choice is motivated in frameworks where the computation of the local second derivatives $\left.\dfrac{\partial^2 f_i}{\partial x_m^2}\right|_{\boldsymbol{x}_i(k)}$ is expensive, or where the second derivatives simply might not be continuous. With this choice Algorithm 2 reduces to a distributed gradient-descent procedure. In fact, for sufficiently small ε, $\boldsymbol{x}_i(k) \approx \overline{\boldsymbol{x}}(\varepsilon k)$ with $\overline{\boldsymbol{x}}(t)$ evolving according to the continuous-time dynamics

$$\dot{\overline{\boldsymbol{x}}}(t) = -\nabla \overline{f}\big(\overline{\boldsymbol{x}}(t)\big) \ ,$$

which one again is guaranteed to converge to the global optimum $\boldsymbol{x}^*$.

The following Table I summarizes the various costs of the previously proposed strategies.

Choice	NRC, Equation (31)	JC, Equation (32)	GDC, Equation (33)
Computational Cost	$O\left(M^3\right)$	$O\left(M\right)$	$O\left(M\right)$
Communication Cost	$O\left(M^2\right)$	$O\left(M\right)$	$O\left(M\right)$
Memory Cost	$O\left(M^2\right)$	$O\left(M\right)$	$O\left(M\right)$

Table I
COMPUTATIONAL, COMMUNICATION AND MEMORY COSTS OF NRC, JC, GDC PER SINGLE UNIT AND SINGLE STEP (LINES 3 TO 5 OF ALGORITHM 2).

The following theorem characterizes the convergence properties of Algorithm 2 (see definitions in page 13) and it is the multidimensional version of Theorem 3:

Theorem 6 Consider Algorithm 2 with arbitrary initial conditions $\boldsymbol{x}_i(0)$, H_i defined in (31) or (32) or (33), and Assumption 1 holding true. Then for every open ball $B_r^{\boldsymbol{x}^*} := \left\{X \in \mathbb{R}^{MN} \mid \|X - \mathbb{1}\otimes\boldsymbol{x}^*\| < r\right\}$ there exist two positive constants $\overline{\varepsilon}_r$, c_r such that if $\varepsilon < \overline{\varepsilon}_r$ then there exists $\gamma_\varepsilon > 0$ s.t., for all k, $X(0) \in B_r^{\boldsymbol{x}^*} \Rightarrow \|X(k) - \mathbb{1}\otimes\boldsymbol{x}^*\| \le c_r e^{-\gamma_\varepsilon k}\|X(0) - \mathbb{1}\otimes\boldsymbol{x}^*\|$.

Proof The proof follows closely the proof of Theorem 3 thus in the interest of space we provide just a simple sketch. Notice that it involves the following alternative notation:

$$\Pi^{\parallel} := \frac{\mathbb{1}\mathbb{1}^T}{N}\otimes I_M, \quad \Pi^{\perp} := \left(I_N - \frac{\mathbb{1}\mathbb{1}^T}{N}\right)\otimes I_M,$$

$$X^{\parallel}(k) := \Pi^{\parallel}X(k), \quad X^{\perp}(k) := \Pi^{\perp}X(k).$$

To prove the theorem we start recognizing that, for sufficiently small ε, the convergence properties of the algorithm are the same as

[3]In centralized approaches, nulling the Hessian's off-diagonal terms is a well-known procedure, see, e.g., [56]. See also [57], [35] for other Jacobi algorithms with different communication structures.

the continuous time system

$$\begin{cases} \varepsilon\dot{V}(t) &= -V(t) + G\big(X(t)\big) \\ \varepsilon\dot{\boldsymbol{W}}(t) &= -\boldsymbol{W}(t) + \boldsymbol{H}\big(X(t)\big) \\ \varepsilon\dot{Y}(t) &= -KY(t) + (I-K)\Big[G\big(X(t)\big) - V(t)\Big] \\ \varepsilon\dot{\boldsymbol{Z}}(t) &= -K\boldsymbol{Z}(t) + (I-K)\Big[\boldsymbol{H}\big(X(t)\big) - \boldsymbol{W}(t)\Big] \\ \dot{\boldsymbol{x}}_i(t) &= -\boldsymbol{x}_i(t) + \big(Z_i(t)\big)^{-1}\boldsymbol{y}_i(t) \qquad i = 1,\dots,N \end{cases} \tag{34}$$

where $K := I_{MN} - (P \otimes I_M)$ (in gray indications for the dimensions of the identity matrices) is again positive semidefinite. Then, with the substitutions $D(t) := Y(t) - V(t)$, $\boldsymbol{B}(t) := \boldsymbol{Z}(t) - \boldsymbol{W}(t)$ one can prove as before that the boundary layer system of (34) admits the globally exponentially stable equilibrium

$$\lim_{t\to\infty} \begin{bmatrix} V(t) \\ \boldsymbol{W}(t) \\ D^{\perp}(t) \\ \boldsymbol{B}^{\perp}(t) \end{bmatrix} = \begin{bmatrix} G(X) \\ \boldsymbol{H}(X) \\ -\Pi^{\perp}G(X) \\ -\Pi^{\perp}\boldsymbol{H}(X) \end{bmatrix}. \tag{35}$$

The stability of the reduced system can instead be analyzed decomposing again its dynamics along the projections given by $\Pi^{\perp}$ and $\Pi^{\parallel}$, obtaining a continuous time non-linear cascade system equivalent to (26) whose global stability properties are ensured by Theorem 1 in [54]. Similarly to the scalar version of the algorithm, the dynamics of the average $\overline{\boldsymbol{x}}(t)$ follow

$$\dot{\overline{\boldsymbol{x}}}(t) = -\overline{\boldsymbol{x}}(t) + \Big(\overline{H}\big(\overline{\boldsymbol{x}}(t)\big)\Big)^{-1}\overline{\boldsymbol{g}}\big(\overline{\boldsymbol{x}}(t)\big)\,. \tag{36}$$

For all the cases (31), (32) and (33), then, it follows that $V(\boldsymbol{x}) := \overline{f}(\boldsymbol{x}) - \overline{f}(\boldsymbol{x}^*)$ is a Lyapunov function for the reduced system which guarantees exponential converges to $\boldsymbol{x}^*$. ■

We remark that $\overline{\varepsilon}_r$ in Theorem 6 depends also on the particular choice for H_i. The list of choices for H_i given by (31), (32) and (33) is not exhaustive. For example, future directions are to implement distributed quasi-Newton procedures. To this regard, we recall that approximations of the Hessians that do not maintain symmetry and positive definiteness or are bad conditioned require additional modification steps, e.g., through Cholesky factorizations [58].

Finally, we notice that in scalar scenarios JC and NRC are equivalent, while GDC corresponds to algorithms requiring just the knowledge of first derivatives.

VI. Numerical Examples

In Section VI-A we analyze the effects of different choices of ε on the scalar NRC on regular graphs and exponential cost functions. We then propose two machine learning problems in Section VI-B, used in Sections VI-C and VI-D, and numerically compare the convergence performance of the NRC, JC, GDC algorithms and other distributed convex optimization algorithms on random geometric graphs.

A. *Effects of the choice of* ε

Consider a ring network of $S = 30$ agents that communicate only to their left and right neighbors through the consensus matrix

$$P = \begin{bmatrix} 0.5 & 0.25 & & & 0.25 \\ 0.25 & 0.5 & 0.25 & & \\ & \ddots & \ddots & \ddots & \\ & & 0.25 & 0.5 & 0.25 \\ 0.25 & & & 0.25 & 0.5 \end{bmatrix}, \tag{37}$$

so that the spectral radius $\rho(P) \approx 0.99$, implying a spectral gap $\sigma(P) \approx 0.01$. Consider also scalar costs of the form $f_i(x) = c_i e^{a_i x} + d_i e^{-b_i x}$, $i = 1,\dots,N$, with $a_i, b_i \sim \mathcal{U}[0, 0.2]$, $c_i, d_i \sim \mathcal{U}[0, 1]$ and where $\mathcal{U}$ indicates the uniform distribution.

Figure 1 compares the evolution of the local states x_i of the continuous system (14) for different values of ε. When ε is not sufficiently small, then the trajectories of $x_i(t)$ are different even if they all start from the same initial condition $x_i(0) = 0$. As ε decreases, the difference between the two time scales becomes more evident and all the trajectories $x_i(k)$ become closer to the trajectory given by the slow NR dynamics $\overline{x}(\varepsilon k)$ given in (15) and guaranteed to converge to the global optimum x^*.

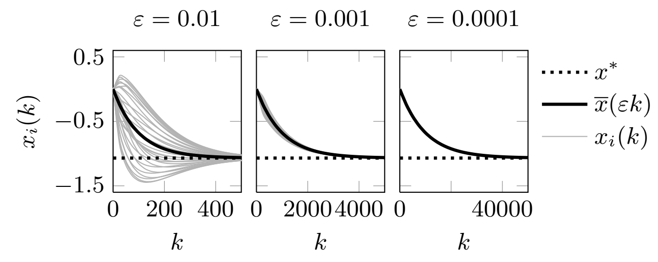
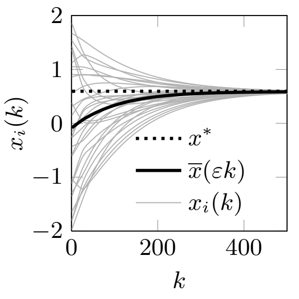
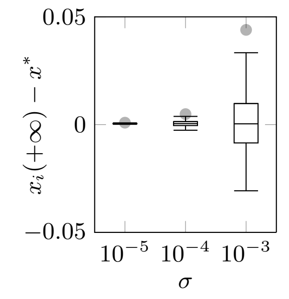


Figure 1. Temporal evolution of system (14) for different values of ε, with $N = 30$. The black dotted line indicates x^*. The black solid line indicates the slow dynamics $\overline{x}(\varepsilon k)$ of Equation (15). As ε decreases. The difference between the time scale of the slow and fast dynamics increases, and the local states $x_i(k)$ converge to the manifold of $\overline{x}(\varepsilon k)$.

In Figure 2 we address the robustness of the proposed algorithm w.r.t. the choice of the initial conditions. In particular, Figure 2(a) shows that if $\alpha = \beta = 0$ then the local states $x_i(t)$ converge to the optimum x^* for arbitrary initial conditions $x_i(0)$, as guaranteed by Theorem 3. Figure 2(b) considers, besides different initial conditions $x_i(0)$, also perturbed initial conditions $\boldsymbol{v}(0)$, $\boldsymbol{w}(0)$, $\boldsymbol{y}(0)$, $\boldsymbol{z}(0)$ leading to non null α's and β's. More precisely we apply Algorithm 1 to different random initial conditions s.t. $\alpha, \beta \sim \mathcal{U}[-\sigma, \sigma]$. Figure 2(b) shows the boxplots of the errors $x_i(+\infty) - x^*$ for different σ's based on 300 Monte Carlo runs with $\varepsilon = 0.01$ and $N = 30$.



(a) Time evolution of the local states $x_i(k)$ with $\boldsymbol{v}(0) = \boldsymbol{w}(0) = \boldsymbol{y}(0) = \boldsymbol{z}(0) = \boldsymbol{0}$ and $x_i(0) \sim \mathcal{U}[-2, 2]$.



(b) Empirical distribution of the errors $x_i(+\infty) - x^*$ under artificially perturbed initial conditions $\alpha(0), \beta(0) \sim \mathcal{U}[-\sigma, \sigma]$ for different values of σ.

Figure 2. Characterization of the dependency of the performance of Algorithm 1 on the initial conditions. In all the experiments $\varepsilon = 0.01$ and $N = 30$.

B. *Optimization problems*

The first problem considered is the distributed training of a Binomial-Deviance based classifier, to be used, e.g., for spam-nonspam classification tasks [59, Chap. 10.5]. More precisely, we consider a database of emails E, where j is the email index, $y_j = -1, 1$ denotes if the email j is considered spam or not, $\boldsymbol{\chi}_j \in \mathbb{R}^M$ numerically summarizes the M features of the j-th email (how many times the words "money", "dollars", etc., appear). If the E emails come from different users that do not want to disclose their private information, then it is meaningful to solve exploiting the distributed optimization algorithms described in the previous sections. More specifically, letting $x = (\boldsymbol{x}, x_0) \in \mathbb{R}^M \times \mathbb{R}$ represents a generic classification hyperplane, training a Binomial-Deviance based classifier corresponds to solve a distributed optimization problem where the local cost functions are given by:

$$f_i\left(x\right) := \sum_{j \in E_i} \log\left(1+\exp\left(-y_j\left(\boldsymbol{\chi}_j^T \boldsymbol{x} + x_0\right)\right)\right) + \gamma \left\|\boldsymbol{x}\right\|_2^2. \quad (38)$$

where E_i is the set of emails available to agent i, $E = \cup_{i=1}^N E_i$, and γ is a global regularization parameter. In the following numerical experiments we consider $|E| = 5000$ emails from the spam-nonspam UCI repository, available at `http://archive.ics.uci.edu/ml/datasets/Spambase`, randomly assigned to 30 different users communicating as in graph of Figure 3. For each email we consider $M = 3$ features (the frequency of words "make", "address", "all") so that the corresponding optimization problem is 4-dimensional.

The second problem considered is a regression problem inspired by the UCI Housing dataset available at `http://archive.ics.uci.edu/ml/datasets/Housing`. In this task, an example $\boldsymbol{\chi}_j \in \mathbb{R}^{M-1}$ is vector representing some features of a house (e.g., per capita crime rate by town, index of accessibility to radial highways, etc.), and $y_j \in \mathbb{R}$ denotes the corresponding median monetary value of of the house. The objective is to obtain a predictor of house value based on these data. Similarly as the previous example, if the datasets come from different users that do not want to disclose their private information, then it is meaningful to solve exploiting the distributed optimization algorithms. This problem can be formulated as a convex regression problem on the following local cost functions:

$$f_i\left(x\right) := \sum_{j \in E_i} \frac{\left(y_j - \boldsymbol{\chi}_j^T \boldsymbol{x} - x_0\right)^2}{\left|y_j - \boldsymbol{\chi}_j^T \boldsymbol{x} - x_0\right| + \beta} + \gamma \left\|\boldsymbol{x}\right\|_2^2. \quad (39)$$

where $x = (\boldsymbol{x}, x_0^*) \in \mathbb{R}^{M-1} \times \mathbb{R}$ is the vector of coefficient for the linear predictor $\hat{y} = \boldsymbol{\chi}^T \boldsymbol{x} + x_0$ and γ is a common regularization parameter. The loss function $\frac{(\cdot)^2}{|\cdot|+\beta}$ corresponds to a smooth version of the Huber robust loss (being $\mathcal{C}^2$) which is employed to minimize the effects of outliers.

The loss function $\frac{(\cdot)^2}{|\cdot|+\beta}$ corresponds to a smooth version of the Huber robust loss (being $\mathcal{C}^2$), with β a parameter dictating for which arguments the loss is pseudo-linear or pseudo-quadratic and to be manually chosen to minimize the effects of outliers. In our experiments we used $M = 4$ features, $\beta = 50$, $\gamma = 1$, and $|E| = 506$ total number of examples in the dataset randomly assigned to the $N = 30$ users communicating as in the graph of Figure 3.

In both the previous problems the optimum, in the following indicated for simplicity with $\boldsymbol{x}^*$, has been computed with a centralized NR with the termination rule "stop when in the last 5 steps the norm of the guessed $\boldsymbol{x}^*$ changed less than $10^{-9}\%$".

C. Comparison of the NRC, JC and GDC algorithms

In Figure 4 we analyze the performance of the three proposed NRC, JC and GDC algorithms defined by Equations (31), (32) or (33) in Algorithm 2 in terms of the relative Mean Square Error (MSE)

$$\mathrm{MSE}\left(k\right) := \frac{1}{N} \sum_{i=1}^{N} \|\boldsymbol{x}_i(k) - \boldsymbol{x}^*\|^2 / \|\boldsymbol{x}^*\|^2$$

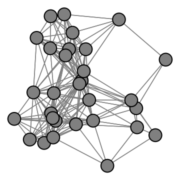
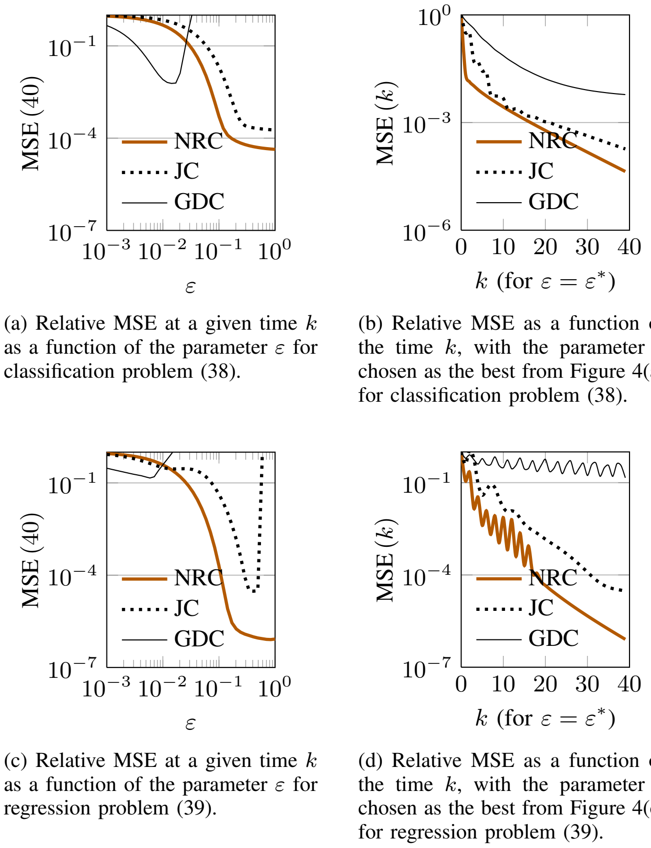


Figure 3. Random geometric graph exploited in the simulations relative to the optimization problem (38). For this graph $\rho(P) \approx 0.9338$, with P the matrix of Metropolis weights.

for the classification and regression optimization problem described above. The consensus matrix P has been by selecting the Metropolis-Hastings weights which are consistent with the communication graph [60]. Panels 4(a) and 4(c) report the MSE obtained at a specific iteration ($k = 40$) by the various algorithms, as a function of ε. These plots thus inspect the sensitivity w.r.t. the choice of the tuning parameters. Consistently with the theorems in the previous section, the GDC and JC algorithms are stable only for ε sufficiently small, while NRC exhibit much larger robustness and best performance for $\varepsilon = 1$. Panels 4(b) and 4(d) instead report the evolutions of the relative MSE as a function of the number of iterations k for the optimally tuned algorithms.



(a) Relative MSE at a given time k as a function of the parameter ε for classification problem (38).

(b) Relative MSE as a function of the time k, with the parameter ε chosen as the best from Figure 4(a) for classification problem (38).

(c) Relative MSE at a given time k as a function of the parameter ε for regression problem (39).

(d) Relative MSE as a function of the time k, with the parameter ε chosen as the best from Figure 4(c) for regression problem (39).

Figure 4. Convergence properties of Algorithm 2 for the problems described in Section VI-B and for different choices of $H_i(\cdot)$. Choice (31) corresponds to the NRC algorithm, (32) to the JC, (33) to the GDC.

We notice that the differences between NRC and JC are evident but not resounding, due to the fact that the Jacobi approximations are in this case a good approximation of the analytical Hessians. Conversely, GDC presents a slower convergence rate which is a known drawback of gradient descent algorithms.

D. Comparisons with other distributed convex optimization algorithms

We now compare Algorithm 1 and its accelerated version, referred as Fast Newton-Raphson Consensus (FNRC) and described in detail below in Algorithm 3), with three popular distributed convex optimization methods, namely the DSM, the Distributed Control Method (DCM) and the ADMM, described respectively in Algorithm 4, 5 and 6. The following discussion provides some details about these strategies.

• FNRC is an accelerated version of Algorithm 1 that inherits the structure of the so called *second order diffusive schedules*, see, e.g., [61], and exploits an additional level of memory to speed up the convergence properties of the consensus strategy. Here the weights multiplying the $\boldsymbol{g}_i$'s and H_i's are necessary to guarantee exact tracking of the current average, i.e. $\sum_i \boldsymbol{y}_i(k) = \sum_i \boldsymbol{g}_i\big(\boldsymbol{x}(k-1)\big)$ for all k. As suggested in [61], we set the φ that weights the gradient and the memory to $\varphi = \dfrac{2}{1+\sqrt{1-\rho(P)^2}}$. This guarantees second order diffusive schedules to be faster than first order ones (even if this does not automatically imply the FNRC to be faster than the NRC). This setting can be considered a valid heuristic to be used when $\rho(P)$ is known. For the graph in Figure 3, $\varphi \approx 1.4730$.

Algorithm 3 Fast Newton-Raphson Consensus

```
1: storage allocation, constraints on the parameters and initialization
   as in Algorithm 1
2: for $k = 1, 2, \ldots$ do
3:     for $i = 1, \ldots, N$ do
4:         $\boldsymbol{x}_i(k) = (1-\varepsilon)\boldsymbol{x}_i(k-1) + \varepsilon\big(Z_i(k-1)\big)^{-1}\boldsymbol{y}_i(k-1)$
5:         $\widetilde{\boldsymbol{y}}_i(k) = \boldsymbol{y}_i(k-1) + \frac{1}{\varphi}\boldsymbol{g}_i\big(\boldsymbol{x}_i(k-1)\big) - \boldsymbol{g}_i\big(\boldsymbol{x}_i(k-2)\big) - \frac{1-\varphi}{\varphi}\boldsymbol{g}_i\big(\boldsymbol{x}_i(k-3)\big)$
6:         $\widetilde{Z}_i(k) = Z_i(k-1) + \frac{1}{\varphi}H_i\big(\boldsymbol{x}_i(k-1)\big) - H_i\big(\boldsymbol{x}_i(k-2)\big) - \frac{1-\varphi}{\varphi}H_i\big(\boldsymbol{x}_i(k-3)\big)$
7:     end for
8:     for $i = 1, \ldots, N$ do
9:         $\boldsymbol{y}_i(k) = \varphi\left(\sum_{j=1}^N p_{ij}\widetilde{\boldsymbol{y}}_j(k)\right) + (1-\varphi)\boldsymbol{y}_i(k-2)$
10:        $\boldsymbol{Z}_i(k) = \varphi\left(\sum_{j=1}^N p_{ij}\widetilde{\boldsymbol{Z}}_j(k)\right) + (1-\varphi)\boldsymbol{Z}_i(k-2)$
11:    end for
12: end for
```

• DSM, as proposed in [29], alternates consensus steps on the current estimated global minimum $\boldsymbol{x}_i(k)$ with subgradient updates of each $\boldsymbol{x}_i(k)$ towards the local minimum. To guarantee the convergence, the amplitude of the local subgradient steps should appropriately decrease. Algorithm 4 presents a synchronous DSM implementation, where ϱ is a tuning parameter and P is the matrix of Metropolis-Hastings weights.

• DCM, as proposed in [41], differentiates from the gradient searching because it forces the states to the global optimum by controlling the subgradient of the global cost. This approach views the subgradient as an input/output map and uses small gain theorems to guarantee the convergence property of the system. Again, each agents i locally computes and exchanges information with its neighbors, collected in the set $\mathcal{N}_i := \{j \mid (i,j) \in \mathcal{E}\}$. DCM is summarized in Algorithm 5, where $\mu, \nu > 0$ are parameters to be designed to ensure the stability property of the system. Specifically, μ is chosen in the interval $0 < \mu < \dfrac{2}{2\max_{i=\{1,\ldots,N\}}|\mathcal{N}_i|+1}$ to bound the induced gain of the subgradients. Also here the parameters have been manually tuned for best convergence rates.

Algorithm 4 DSM [29]

```
   (storage allocation and constraints on parameters)
1: $\boldsymbol{x}_i(k) \in \mathbb{R}^N$ for all $i$. $\varrho \in \mathbb{R}_+$
   (initialization)
2: $\boldsymbol{x}_i(0) = \boldsymbol{0}$
   (main algorithm)
3: for $k = 0, 1, \ldots$ do
4:     for $i = 1, \ldots, N$ do
5:         $\boldsymbol{x}_i(k+1) = \sum_{j=1}^N p_{ij}\left(\boldsymbol{x}_j(k) - \frac{\varrho}{k}\nabla f_j\big(\boldsymbol{x}_j(k)\big)\right)$
6:     end for
7: end for
```

Algorithm 5 DCM [41]

```
   (storage allocation and constraints on parameters)
1: $\boldsymbol{x}_i(k), \boldsymbol{z}_i(k) \in \mathbb{R}^M$, for all $i$. $\mu, \nu \in \mathbb{R}_+$
   (initialization)
2: $\boldsymbol{x}_i(0) = \boldsymbol{z}_i(0) = \boldsymbol{0}$ for all $i$
   (main algorithm)
3: for $k = 0, 1, \ldots$ do
4:     for $i = 1, \ldots, N$ do
5:         $\boldsymbol{z}_i(k+1) = \boldsymbol{z}_i(k) + \mu\sum_{j\in\mathcal{N}_i}\big(\boldsymbol{x}_i(k) - \boldsymbol{x}_j(k)\big)$
6:         $\boldsymbol{x}_i(k+1) = \boldsymbol{x}_i(k) + \mu\sum_{j\in\mathcal{N}_i}\big(\boldsymbol{x}_j(k) - \boldsymbol{x}_i(k)\big) + \mu\sum_{j\in\mathcal{N}_i}\big(\boldsymbol{z}_j(k) - \boldsymbol{z}_i(k)\big) - \mu\nu\nabla f_i\big(\boldsymbol{x}_i(k)\big)$
7:     end for
8: end for
```

• ADMM, instead, requires the augmentation of the system through additional constraints that do not change the optimal solution but allow the Lagrangian formalism. There exist different implementations of ADMM in distributed contexts, see, e.g., [7], [62], [12, pp. 253-261]. For simplicity we consider the following formulation,

$$\min_{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_N} \sum_{i=1}^N f_i(\boldsymbol{x}_i) \quad \text{s.t. } \boldsymbol{z}_{(i,j)} = \boldsymbol{x}_i, \quad \forall i \in \mathcal{N}, \quad \forall (i,j) \in \mathcal{E},$$

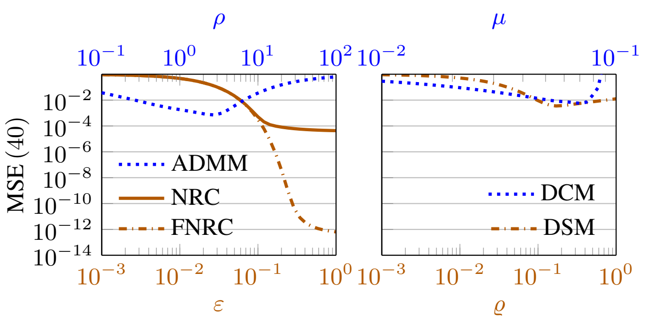
where the auxiliary variables $\boldsymbol{z}_{(i,j)}$ correspond to the different links in the network, and where the local Augmented Lagrangian is given by

$$L_i(\boldsymbol{x}_i, k) := f_i(\boldsymbol{x}_i) + \sum_{j\in\mathcal{N}_i}\boldsymbol{y}_{(i,j)}\big(\boldsymbol{x}_i - \boldsymbol{z}_{(i,j)}\big) + \sum_{j\in\mathcal{N}_i}\frac{\delta}{2}\left\|\boldsymbol{x}_i - \boldsymbol{z}_{(i,j)}\right\|^2,$$

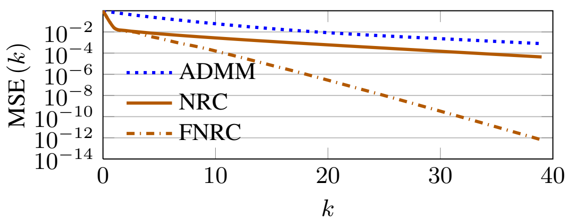
with δ a tuning parameter (see [63] for a discussion on how to tune it) and the $\boldsymbol{y}_{(i,j)}$'s Lagrange multipliers.

Figure 5 then compares the previously cited algorithms as did in Figure 4. The first panel thus reports the relative MSE of the various algorithms at a given number of iterations ($k = 40$) as a function of the parameters. The second panel instead reports the temporal evolution of the relative MSE for the case of optimal tuning.

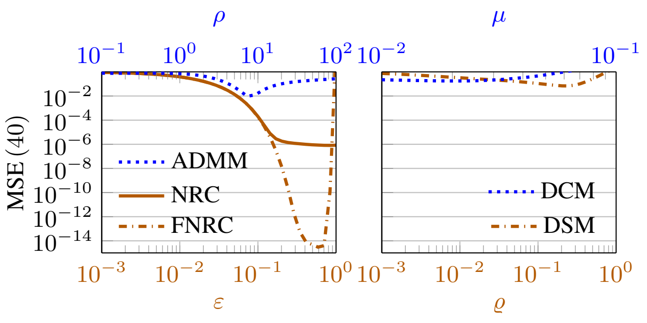
We notice that the DCM and the DSM are both much slower than the NRC, FNRC and ADMM. Moreover, both the NRC and its accelerated version converge faster than the ADMM, even if not tuned at their best. These numerical examples seem to indicate that the proposed NRC might be a viable alternative to the ADMM, although further comparisons are needed to strengthen this claim. Moreover, a substantial potential advantage of NRC as compared to ADMM is that the former can be readily adapted to asynchronous and time-varying graphs, as preliminary made in [64]. Moreover, as in the



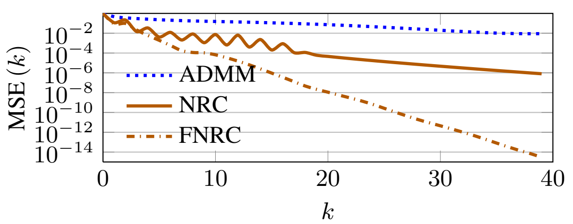
(a) Relative MSE at a given time k as a function of the algorithms parameters for problem (38). For the DCM, $\nu = 1.7$.



(b) Relative MSE as a function of the time k for the three fastest algorithms for problem (38). Their parameters are chosen as the best ones from Figure 5(a).



(c) Relative MSE at a given time k as a function of the algorithms parameters for problem (39). For the DCM, $\nu = 1.7$.



(d) Relative MSE as a function of the time k for the three fastest algorithms for problem (39). Their parameters are chosen as the best ones from Figure 5(c).

Figure 5. Convergence properties of Algorithm 2 for the problems described in Section VI-B.

Algorithm 6 ADMM [7, pp. 253-261]

```
    (storage allocation and constraints on parameters)
 1: $\boldsymbol{x}_i(k), \boldsymbol{z}_{(i,j)}(k), \boldsymbol{y}_{(i,j)}(k) \in \mathbb{R}^M$, $\delta \in (0,1)$
    (initialization)
 2: $\boldsymbol{x}_i(k) = \boldsymbol{z}_{(i,j)}(k) = \boldsymbol{y}_{(i,j)}(k) = \mathbf{0}$
    (main algorithm)
 3: for $k = 0, 1, \ldots$ do
 4:     for $i = 1, \ldots, N$ do
 5:         $\boldsymbol{x}_i(k+1) = \arg\min_{\boldsymbol{x}_i} L_i(\boldsymbol{x}_i, k)$
 6:         for $j \in \mathcal{N}_i$ do
 7:             $\boldsymbol{z}_{(i,j)}(k+1) = \frac{1}{2\delta}\Big(\boldsymbol{y}_{(i,j)}(k) + \boldsymbol{y}_{(j,i)}(k)\Big) + \frac{1}{2}\Big(\boldsymbol{x}_i(k+1) + \boldsymbol{x}_j(k+1)\Big)$
 8:             $\boldsymbol{y}_{(i,j)}(k+1) = \boldsymbol{y}_{(i,j)}(k) + \delta\Big(\boldsymbol{x}_i(k+1) - \boldsymbol{z}_{(i,j)}(k+1)\Big)$
 9:         end for
10:     end for
11: end for
```

case of the FNRC, the strategy can implement any improved linear consensus algorithm.

VII. Conclusion

We proposed a novel distributed optimization strategy suitable for convex, unconstrained, multidimensional, smooth and separable cost functions. The algorithm does not rely on Lagrangian formalisms and acts as a distributed Newton-Raphson optimization strategy by repeating the following steps: agents first locally compute and update second order Taylor expansions around the current local guesses and then they suitably combine them by means of average consensus algorithms to obtain a sort of approximated Taylor expansion of the global cost. This allows each agent to infer a local Newton direction, used to locally update the guess of the global minimum.

Importantly, the average consensus protocols and the local updates steps have different time-scales, and the whole algorithm is proved to be convergent only if the stepsize is sufficiently slow. Numerical simulations show that, if suitably tuned, the algorithm is generally faster than DSMs and competitive with respect to ADMMs.

The set of open research paths is extremely vast. We envisage three main avenues. The first one is to study how the agents can dynamically and locally tune the speed of the local updates w.r.t. the consensus process, namely how to tune their local step-size ε_i. In fact large values of ε gives faster convergence but might lead to instability. A second one is to let the communication protocol be asynchronous: in this regard we notice that some preliminary attempts can be found in [64]. A final branch is about the analytical characterization of the rate of convergence of the proposed strategies and the extensions to non-smooth convex functions.

References

[1] F. Zanella, D. Varagnolo, A. Cenedese, G. Pillonetto, and L. Schenato, "Newton-Raphson consensus for distributed convex optimization," in *IEEE Conference on Decision and Control and European Control Conference*, Dec. 2011, pp. 5917–5922.

[2] ——, "Multidimensional Newton-Raphson consensus for distributed convex optimization," in *American Control Conference*, 2012.

[3] N. Z. Shor, *Minimization Methods for Non-Differentiable Functions*. Springer-Verlag, 1985.

[4] D. P. Bertsekas, A. Nedić, and A. E. Ozdaglar, *Convex Analysis and Optimization*. Athena Scientific, 2003.

[5] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[6] J. N. Tsitsiklis, “Problems in decentralized decision making and computation,” Ph.D. dissertation, Massachusetts Institute of Technology, 1984.

[7] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, 1997.

[8] D. P. Bertsekas, *Network Optimization: Continuous and Discrete Models*. Belmont, Massachusetts: Athena Scientific, 1998.

[9] M. Bürger, G. Notarstefano, F. Bullo, and F. Allgöwer, “A distributed simplex algorithm for degenerate linear programs and multi-agent assignments,” *Automatica*, vol. 48, no. 9, pp. 2298 – 2304, 2012.

[10] D. P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods*. Boston, MA: Academic Press, 1982.

[11] M. R. Hestenes, “Multiplier and gradient methods,” *Journal of Optimization Theory and Applications*, vol. 4, no. 5, pp. 303–320, 1969.

[12] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, “Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers,” Stanford Statistics Dept., Tech. Rep., 2010.

[13] T. Erseghe, D. Zennaro, E. Dall’Anese, and L. Vangelista, “Fast Consensus by the Alternating Direction Multipliers Method,” *IEEE Transactions on Signal Processing*, vol. 59, no. 11, pp. 5523–5537, Nov. 2011.

[14] B. He and X. Yuan, “On the O(1/t) convergence rate of alternating direction method,” *SIAM Journal on Numerical Analysis (to appear)*, 2011.

[15] E. Wei and A. Ozdaglar, “Distributed Alternating Direction Method of Multipliers,” in *IEEE Conference on Decision and Control*, 2012.

[16] J. a. Mota, J. Xavier, P. Aguiar, and M. Püschel, “Distributed ADMM for Model Predictive Control and Congestion Control,” in *IEEE Conference on Decision and Control*, 2012.

[17] D. Jakovetić, J. a. Xavier, and J. M. F. Moura, “Cooperative convex optimization in networked systems: Augmented lagrangian algorithms with directed gossip communication,” *IEEE Transactions on Signal Processing*, vol. 59, no. 8, pp. 3889 – 3902, Aug. 2011.

[18] V. F. Dem’yanov and L. V. Vasil’ev, *Nondifferentiable Optimization*. Springer - Verlag, 1985.

[19] B. Johansson, “On Distributed Optimization in Networked Systems,” Ph.D. dissertation, KTH Royal Institute of Technology, 2008.

[20] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, “Randomized Gossip Algorithms,” *IEEE Transactions on Information Theory / ACM Transactions on Networking*, vol. 52, no. 6, pp. 2508–2530, June 2006.

[21] A. Ribeiro, “Ergodic stochastic optimization algorithms for wireless communication and networking,” *IEEE Transactions on Signal Processing*, vol. 58, no. 12, pp. 6369 – 6386, Dec. 2010.

[22] A. Nedić and D. P. Bertsekas, “Incremental subgradient methods for nondifferentiable optimization,” *SIAM Journal on Optimization*, vol. 12, no. 1, pp. 109–138, 2001.

[23] A. Nedić, D. Bertsekas, and V. Borkar, “Distributed asynchronous incremental subgradient methods,” *Studies in Computational Mathematics*, vol. 8, pp. 381–407, 2001.

[24] A. Nedić and A. Ozdaglar, “Approximate primal solutions and rate analysis for dual subgradient methods,” *SIAM Journal on Optimization*, vol. 19, no. 4, pp. 1757 – 1780, 2008.

[25] K. C. Kiwiel, “Convergence of approximate and incremental subgradient methods for convex optimization,” *SIAM Journal on Optimization*, vol. 14, no. 3, pp. 807–840, 2004.

[26] D. Blatt, A. Hero, and H. Gauchman, “A convergent incremental gradient method with a constant step size,” *SIAM Journal on Optimization*, vol. 18, no. 1, pp. 29–51, 2007.

[27] L. Xiao and S. Boyd, “Optimal scaling of a gradient method for distributed resource allocation,” *Journal of optimization theory and applications*, vol. 129, no. 3, pp. 469 – 488, 2006.

[28] S. S. Ram, A. Nedić, and V. V. Veeravalli, “Incremental stochastic subgradient algorithms for convex optimzation,” *SIAM Journal on Optimization*, vol. 20, no. 2, pp. 691–717, 2009.

[29] A. Nedić and A. Ozdaglar, “Distributed Subgradient Methods for Multi-Agent Optimization,” *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, 2009.

[30] B. Johansson, M. Rabi, and M. Johansson, “A randomized incremental subgradient method for distributed optimization in networked systems,” *SIAM Journal on Optimization*, vol. 20, no. 3, pp. 1157–1170, 2009.

[31] I. Lobel, A. Ozdaglar, and D. Feijer, “Distributed multi-agent optimization with state-dependent communication,” *Mathematical Programming*, vol. 129, no. 2, pp. 255 – 284, 2011.

[32] A. Nedić, “Asynchronous Broadcast-Based Convex Optimization over a Network,” *IEEE Transactions on Automatic Control*, vol. 56, no. 6, pp. 1337 – 1351, June 2010.

[33] E. Ghadimi, I. Shames, and M. Johansson, “Accelerated Gradient Methods for Networked Optimization,” *IEEE Transactions on Signal Processing (under review)*, 2012.

[34] A. Jadbabaie, A. Ozdaglar, and M. Zargham, “A Distributed Newton Method for Network Optimization,” in *IEEE Conference on Decision and Control*. IEEE, 2009, pp. 2736–2741.

[35] M. Zargham, A. Ribeiro, A. Ozdaglar, and A. Jadbabaie, “Accelerated Dual Descent for Network Optimization,” in *American Control Conference*, 2011.

[36] E. Wei, A. Ozdaglar, and A. Jadbabaie, “A Distributed Newton Method for Network Utility Maximization,” in *IEEE Conference on Decision and Control*, 2010, pp. 1816 – 1821.

[37] R. S. Dembo, S. C. Eisenstat, and T. Steihaug, “Inexact newton methods,” *SIAM Journal on Numerical*, vol. 19, no. 2, pp. 400–408, 1982.

[38] A. Nedić, A. Ozdaglar, and P. A. Parrilo, “Constrained Consensus and Optimization in Multi-Agent Networks,” *IEEE Transactions on Automatic Control*, vol. 55, no. 4, pp. 922–938, Apr. 2010.

[39] M. Zhu and S. Martínez, “On Distributed Convex Optimization Under Inequality and Equality Constraints,” *IEEE Transactions on Automatic Control*, vol. 57, no. 1, pp. 151–164, 2012.

[40] C. Fischione, “F-Lipschitz Optimization with Wireless Sensor Networks Applications,” *IEEE Transactions on Automatic Control*, vol. 56, no. 10, pp. 2319 – 2331, 2011.

[41] J. Wang and N. Elia, “Control approach to distributed optimization,” in *Forty-Eighth Annual Allerton Conference*, vol. 1, no. 1. Allerton, Illinois, USA: IEEE, Sept. 2010, pp. 557–561.

[42] N. Freris and A. Zouzias, “Fast Distributed Smoothing for Network Clock Synchronization,” in *IEEE Conference on Decision and Control*, 2012.

[43] F. Garin and L. Schenato, *A survey on distributed estimation and control applications using linear consensus algorithms*. Springer, 2011, vol. 406, ch. 3, pp. 75–107.

[44] H. K. Khalil, *Nonlinear Systems*, 3rd ed. Prentice Hall, 2001.

[45] F. Fagnani and S. Zampieri, “Randomized consensus algorithms over large scale networks,” *IEEE Journal on Selected Areas in Communications*, vol. 26, no. 4, pp. 634–649, May 2008.

[46] A. D. Domínguez-García, C. N. Hadjicostis, and N. H. Vaidya, “Distributed Algorithms for Consensus and Coordination in the Presence of Packet-Dropping Communication Links Part I : Statistical Moments Analysis Approach,” Coordinated Sciences Laboratory, University of Illinois at Urbana-Champaign, Tech. Rep., 2011.

[47] Y. C. Ho, L. Servi, and R. Suri, “A class of center-free resource allocation algorithms,” *Large Scale Systems*, vol. 1, no. 1, pp. 51 – 62, 1980.

[48] K. Tanabe, “Global analysis of continuous analogues of the Levenberg-Marquardt and Newton-Raphson methods for solving nonlinear equations,” *Annals of the Institute of Statistical Mathematics*, vol. 37, no. 1, pp. 189–203, 1985.

[49] R. Hauser and J. Nedić, “The Continuous Newton-Raphson Method Can Look Ahead,” *SIAM Journal on Optimization*, vol. 15, pp. 915–925, 2005.

[50] P. Kokotović, H. K. Khalil, and J. O’Reilly, *Singular Perturbation Methods in Control: Analysis and Design*, ser. Classics in applied mathematics. SIAM, 1999, no. 25.

[51] L. Xiao, S. Boyd, and S. Lall, “A Scheme for Robust Distributed Sensor Fusion Based on Average Consensus,” in *International symposium on Information processing in sensor networks*, 2005, pp. 63–70.

[52] S. Bolognani, S. Del Favero, L. Schenato, and D. Varagnolo, “Consensus-based distributed sensor calibration and least-square parameter identification in WSNs,” *International Journal of Robust and Nonlinear Control*, vol. 20, no. 2, pp. 176–193, Jan. 2010.

[53] A. R. Teel, D. Nesic, and P. V. Kokotovic, “A note on input-to-state stability of sampled-data nonlinear systems,” in *IEEE Conference on Decision and Control*, vol. 3, Dec. 1998, pp. 2473–2478.

[54] V. Sundarapandian, “Global asymptotic stability of nonlinear cascade systems,” *Applied Mathematics Letters*, vol. 15, no. 3, pp. 275–277, Apr. 2002.

[55] T. Sahai, A. Speranzon, and A. Banaszuk, “Hearing the clusters of a graph: A distributed algorithm,” *Automatica*, vol. 48, no. 1, pp. 15–24, Jan. 2012.

[56] S. Becker and Y. Le Cun, “Improving the convergence of back-propagation learning with second order methods,” University of Toronto, Tech. Rep., Sept. 1988.

[57] S. Athuraliya and S. H. Low, “Optimization flow control with Newton like algorithm,” *Telecommunication Systems*, vol. 15, pp. 345–358, 2000.

[58] G. H. Golub and C. F. Van Loan, *Matrix computations*, 3rd ed. John Hopkins University Press, 1996.

[59] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed. New York: Springer, 2001.

[60] L. Xiao, S. Boyd, and S.-J. Kim, “Distributed average consensus with least-mean-square deviation,” *Journal of Parallel and Distributed Computing*, vol. 67, no. 1, pp. 33–46, Jan. 2007.

[61] S. Muthukrishnan, B. Ghosh, and M. H. Schultz, “First and Second Order Diffusive Methods for Rapid, Coarse, Distributed Load Balancing,” *Theory of Computing Systems*, vol. 31, no. 4, pp. 331 – 354, 1998.

[62] I. D. Schizas, A. Ribeiro, and G. B. Giannakis, “Consensus in Ad Hoc WSNs With Noisy Links - Part I: Distributed Estimation of Deterministic Signals,” *IEEE Transactions on Signal Processing*, vol. 56, pp. 350–364, Jan. 2008.

[63] E. Ghadimi, A. Teixeira, I. Shames, and M. Johansson, “On the Optimal Step-size Selection for the Alternating Direction Method of Multipliers,” in *Necsys*, 2012.

[64] F. Zanella, D. Varagnolo, A. Cenedese, G. Pillonetto, and L. Schenato, “Asynchronous Newton-Raphson Consensus for Distributed Convex Optimization,” in *Necsys 2012*, 2012.

Appendix - Additional notation

(scalar case)

$$\begin{aligned}
g_i\big(x_i(k)\big) &:= f_i''\big(x_i(k)\big)x_i(k) - f_i'\big(x_i(k)\big) \\
h_i\big(x_i(k)\big) &:= f_i''\big(x_i(k)\big) \\
\boldsymbol{x}(k) &:= [x_1(k) \ \cdots \ x_N(k)]^T \\
\boldsymbol{y}(k) &:= [y_1(k) \ \cdots \ y_N(k)]^T \\
\boldsymbol{z}(k) &:= [z_1(k) \ \cdots \ z_N(k)]^T \\
\boldsymbol{g}\big(\boldsymbol{x}(k)\big) &:= \big[g_1\big(x_1(k)\big) \ \cdots \ g_N\big(x_N(k)\big)\big]^T \\
\boldsymbol{h}\big(\boldsymbol{x}(k)\big) &:= \big[h_1\big(x_1(k)\big) \ \cdots \ h_N\big(x_N(k)\big)\big]^T
\end{aligned}$$

(vectorial case)

$$\begin{aligned}
\boldsymbol{x}_i(k) &:= [x_{i,1}(k) \ \cdots \ x_{i,M}(k)]^T \in \mathbb{R}^M \\
X(k) &:= \Big[\boldsymbol{x}_1(k)^T \ \cdots \ \boldsymbol{x}_N(k)^T\Big]^T \in \mathbb{R}^{MN} \\
\nabla f\big(\boldsymbol{x}_i(k)\big) &:= \left[\left.\frac{\partial f}{\partial x_1}\right|_{\boldsymbol{x}_i(k)} \ \cdots \ \left.\frac{\partial f}{\partial x_M}\right|_{\boldsymbol{x}_i(k)} \right]^T \in \mathbb{R}^M \\
\nabla^2 f\left(\boldsymbol{x}_i(k)\right) &:= \begin{bmatrix} \ddots & \vdots & \\ \cdots & \left.\dfrac{\partial^2 f}{\partial x_m \partial x_n}\right|_{\boldsymbol{x}_i(k)} & \cdots \\ & \vdots & \ddots \end{bmatrix} \in \mathbb{R}^{M \times M} \\
Y(k) &:= \Big[\boldsymbol{y}_1(k)^T \ \cdots \ \boldsymbol{y}_N(k)^T \ \Big]^T \in \mathbb{R}^{MN} \\
\boldsymbol{Z}(k) &:= \Big[Z_1(k)^T \ \cdots \ Z_N(k)^T \ \Big]^T \in \mathbb{R}^{MN \times M} \\
H_i\big(\boldsymbol{x}_i(k)\big) &:= \textit{Equations } (31) \textit{ or } (32) \textit{ or } (33), \in \mathbb{R}^{M \times M} \\
\boldsymbol{H}\big(X(k)\big) &:= \Big[H_1\big(\boldsymbol{x}_1(k)\big)^T \ \cdots \ H_N\big(\boldsymbol{x}_N(k)\big)^T\Big]^T \in \mathbb{R}^{MN \times M} \\
\boldsymbol{g}_i\big(\boldsymbol{x}_i(k)\big) &:= H_i\big(\boldsymbol{x}_i(k)\big)\boldsymbol{x}_i(k) - \nabla f_i\big(\boldsymbol{x}_i(k)\big) \in \mathbb{R}^M \\
G\big(X(k)\big) &:= \Big[\boldsymbol{g}_1\big(\boldsymbol{x}_1(k)\big)^T \ \cdots \ \boldsymbol{g}_N\big(\boldsymbol{x}_N(k)\big)^T\Big] \in \mathbb{R}^{MN}
\end{aligned}$$