Totally Asynchronous Distributed Quadratic Programming with Independent Stepsizes and Regularizations

Matthew Ubl* and Matthew T. Hale*

Abstract

Quadratic programs arise in robotics, communications, smart grids, and many other applications. As these problems grow in size, finding solutions becomes much more computationally demanding, and new algorithms are needed to efficiently solve them. Targeting large-scale problems, we develop a multi-agent quadratic programming framework in which each agent updates only a small number of the total decision variables in a problem. Agents communicate their updated values to each other, though we do not impose any restrictions on the timing with which they do so, nor on the delays in these transmissions. Furthermore, we allow weak parametric coupling among agents, in the sense that they are free to independently choose their stepsizes, subject to mild restrictions. We show that these stepsize restrictions depend upon a problem's condition number. We further provide the means for agents to independently regularize the problem they solve, thereby improving condition numbers and, as we will show, convergence properties, while preserving agents' independence in selecting parameters. Simulation results are provided to demonstrate the success of this framework on a practical quadratic program.

I. INTRODUCTION

Convex optimization problems arise in a diverse array of engineering applications, including signal processing [1], robotics [2], [3], communications [4], machine learning [5], and many others [6]. In all of these areas, problems can become very large as the number of network members (robots, processors, etc.) becomes large. Accordingly, there has arisen interest in solving large-scale optimization problems. A common feature of large-scale solvers is that they are parallelized or distributed among a collection of agents in some way. As the number of agents grows, it can be difficult or impossible to ensure synchrony among distributed computations and communications, and there has therefore arisen interest in distributed asynchronous optimization algorithms.

One line of research considers asynchronous optimization algorithms in which agents' communication topologies vary in time. A representative sample of this work includes [7]–[12], and these algorithms all rely on an underlying averaging-based update law, i.e., different agents update the same decision variables and then repeatedly average their iterates to mitigate disagreements that stem from asynchrony. These approaches (and others in the literature) require some form of graph connectivity over intervals of a finite length. In this paper, we are interested in cases in which delay bounds are outside agents' control, e.g., due to environmental hazards and adversarial jamming

---

*Department of Mechanical and Aerospace Engineering, University of Florida, Gainesville, FL, 32611, USA. Emails: {m.ubl,matthewhale}@ufl.edu

for a team of mobile autonomous agents. In these settings, verifying graph connectivity can be difficult for single agents to do, and it may not be possible to even check that connectivity assumptions are satisfied. Furthermore, even if such checking is possible, it will be difficult to reliably attain connectivity with unreliable and impaired communications. These challenges can cause existing approaches to perform poorly or fail outright. For multi-agent systems with impaired communications, we are interested in developing an algorithmic framework that succeeds without requiring any delay bound assumptions.

In this paper, we develop a totally asynchronous quadratic programming (QP) framework for multi-agent optimization. Our interest in quadratic programming is motivated by problems in robotics [3] and data science [13], where some standard problems can be formalized as QPs. The "totally asynchronous" label originates in [14], and it describes a class of algorithms which tolerate arbitrarily long delays, which our framework will do. In addition, our developments will use block-based update laws in which each agent updates only a small subset of the decision variables in a problem, which reduces each agent's computational burden and, as we will show, reduces its onboard storage requirements as well. Both of these properties help enable the solution of large-scale problems.

Other work on distributed quadratic programming includes [15]–[20]. Our work differs from these existing results because we consider non-separable objective functions, and because we consider unstructured update laws (i.e., we do not require communications and computations to occur in a particular sequence or pattern). Furthermore, we consider only deterministic problems, and our framework converges exactly to a problem's solution, while some existing works consider stochastic problems and converge approximately.

Asynchrony in agents' communications and computations implies that they will send and receive different information at different times. As a result, they will disagree about the values of decision variables in a problem. Just as it is difficult for agents to agree on this information, it can also be difficult to agree on a stepsize value in their algorithms. One could envision a network of agents solving an agreement problem, e.g., [21], to compute a common stepsize, though we instead allow agents to independently choose stepsizes, subject to mild restrictions, thereby eliminating the need to reach agreement before optimizing.

It has been shown that regularizing problems can endow them with an inherent robustness to asynchrony and improved convergence properties, e.g., [22]–[24]. Although regularizing is not required here, we show, in a precise sense, that regularizing improves convergence properties of our framework as well. It is common for regularization-based approaches to require agents to use the same regularization parameter, though this is undesirable for the same reasons as using a common stepsize. Therefore, we allow agents to independently choose regularization parameters as well.

To the best of our knowledge, few works have considered both independent stepsizes and regularizations. The most relevant is [22], which considers primal-dual algorithms for constrained problems and synchronous primal updates. This paper is different in that we consider unconstrained problems with totally asynchronous updates. Regularizing of course introduces errors in a solution, and we bound these errors in terms of agents' regularization parameters. The end result is a totally asynchronous quadratic programming framework in which communication delays can be unbounded, and in which agents independently choose all stepsize and regularization parameters.

The rest of the paper is organized as follows. Section II provides background on QPs and formal problem statements. Then, Section III proposes an update law to solve the problems of interest, and Section IV proves its convergence. Next, Section V shows that independent regularizations lead to better-conditioned problems, and Section VI provides error bounds in terms of agents' regularizations; Section VII provides several corollaries and special cases for these bounds. After that, Section VIII provides simulation results, and Section IX concludes the paper.

II. BACKGROUND AND PROBLEM STATEMENT

In this section, we describe the quadratic optimization problems to be solved, as well as the assumptions imposed upon these problems and the agents that solve them. We then describe agents' stepsizes and regularizations and introduce the need to allow agents to choose these parameters independently. We next describe the benefits of independent regularizations, and give two formal problem statements that will be the focus of the remainder of the paper.

We consider a quadratic optimization problem distributed across a network of N agents, where agents are indexed over $i \in [N] := \{1, ..., N\}$. Agent i has a decision variable $x_i \in \mathbb{R}^{n_i}, n_i \in \mathbb{N}$, which we refer to as its state, and we allow for $n_i \neq n_j$ if $i \neq j$. The state x_i is subject to the set constraint $x_i \in X_i \subset \mathbb{R}^{n_i}$, and we make the following assumption about each X_i.

Assumption 1: For all $i \in [N]$, the set $X_i \subset \mathbb{R}^{n_i}$ is non-empty, compact, and convex. $\triangle$

We define the network-level constraint set $X := X_1 \times \cdots \times X_N$, and Assumption 1 implies that X is non-empty, compact, and convex. We further define the global state as $x := \left(x_1^T, ..., x_N^T\right)^T \in X \subset \mathbb{R}^n$, where $n = \sum_{i \in [N]} n_i$. We consider quadratic objectives

$$f(x) := \frac{1}{2}x^T Q x + r^T x,$$

where $Q \in \mathbb{R}^{n \times n}$ and $r \in \mathbb{R}^n$. We then make the following assumption about f.

Assumption 2: In f, $Q = Q^T \succ 0$. $\triangle$

Because Q is positive definite, f is strongly convex, and because f is quadratic, it is twice continuously differentiable, which we indicate by writing that f is C^2. In addition, $\nabla f = Qx + r$, and ∇f is therefore Lipschitz with constant $\|Q\|_2$. In this paper, we divide $n \times n$ matrices into blocks. Given a matrix $B \in \mathbb{R}^{n \times n}$, where $n = \sum_{i=1}^N n_i$, the ith block of B, denoted $B^{[i]}$, is the $n_i \times n$ matrix formed by rows of B with indices $\sum_{k=1}^{i-1} n_k + 1$ through $\sum_{k=1}^i n_k$. In other words, $B^{[1]}$ is the first n_1 rows of B, $B^{[2]}$ is the next n_2 rows, etc. Similarly, for a vector b, $b^{[1]}$ is the first n_1 entries of b, $b^{[2]}$ is the next n_2 entries, etc. Using this notion of a matrix block, we define $\nabla_i f := \frac{\partial f}{\partial x_i}$, and we see that $\nabla_i f(x) = Q^{[i]} x + r^{[i]}$.

Following our goal of reducing parametric coupling between agents, we wish to allow agents to select stepsizes independently. Allowing independent stepsizes will preclude the need for agents to agree on a single value before optimizing, which gives agents a degree of freedom in their update laws and eliminates the need to solve an agreement problem before optimizing. Bearing this in mind, we state the following problem, which will be one focus of the remainder of the paper.

Problem 1: Design a totally asynchronous distributed optimization algorithm to solve

$$\underset{x \in X}{\text{minimize}} \quad \frac{1}{2}x^T Q x + r^T x,$$

where only agent i updates x_i, and where agents choose stepsizes independently. $\triangle$

While an algorithm that satisfies the conditions stated in Problem 1 is sufficient to find a solution, there are additional characteristics of f that can be considered, namely k_Q, the spectral condition number of Q. For a matrix Q satisfying Assumption 2, with eigenvalues $\lambda_1(Q) \geq \lambda_2(Q) \geq ... \geq \lambda_n(Q)$, k_Q is defined as $k_Q := \frac{\lambda_1(Q)}{\lambda_n(Q)}$. In standard centralized quadratic programming analyses, we find that k_Q plays a vital role in determining convergence rates. In particular, a problem with a large k_Q is described as "ill-conditioned," and it will converge slower than a similar problem with a small k_Q. We will find that this is indeed the case in Problem 1 as well. Additionally, as will be shown below, k_Q restricts agents' choices of stepsizes. Therefore, a reduction in k_Q will lead to a wider range of stepsize choices for agents, in addition to faster convergence.

Regularizations are commonly used for centralized quadratic programs to improve k_Q (i.e., to reduce it), and we will therefore use them here. However, in keeping with the independence of agents' parameters, we wish to allow agents to choose independent regularization parameters as well. In particular, we should allow agent i to use use the regularization parameter $\alpha_i > 0$, while allowing $\alpha_i \neq \alpha_j$ for $i \neq j$. The regularized form of f, denoted f_A, is

$$f_A(x) := f(x) + \frac{1}{2}x^T A x = \frac{1}{2}x^T(Q + A)x + r^T x,$$

where $A = \text{diag}\left(\alpha_1 I_{n_1}, ..., \alpha_N I_{n_N}\right)$, and where I_{n_i} is the $n_i \times n_i$ identity matrix. Note that $\nabla_i f_A = Q^{[i]}x + r^{[i]} + \alpha_i x_i$, where we see that only α_i affects agent i's updates.

With the goal of independent regularizations in mind, we now state the second problem that we will solve.

Problem 2: Design a totally asynchronous distributed optimization algorithm to solve

$$\underset{x \in X}{\text{minimize}} \quad \frac{1}{2}x^T(Q + A)x + r^T x,$$

where only agent i updates x_i, and where agents independently choose their stepsizes and regularizations. $\triangle$

Section III specifies the structure of the asynchronous communications and computations used to solve Problem 1, and we will solve Problem 1 in Section IV. Afterwards, we will solve Problem 2 in Section V.

III. BLOCK-BASED MULTI-AGENT UPDATE LAW

To define the exact update law for each agent's state, we must first describe the information stored onboard each agent and how agents communicate with each other. Each agent will store a vector containing its own state and that of every agent it communicates with. Formally, we will denote agent i's full vector of states by x^i, and this is agent i's local copy of the global state. Agent i's own states in this vector are denoted by x_i^i. The current values stored onboard agent i for agent j's states are denoted by x_j^i. In the forthcoming update law, agent i will only compute updates for x_i^i, and it will share only x_i^i with other agents when communicating. Agent i will only change the value of x_j^i when agent j sends its own state to agent i.

At time k, agent i's full state vector is denoted $x^i(k)$, with its own states denoted $x_i^i(k)$ and those of agent j denoted $x_j^i(k)$. At any timestep, agent i may or may not update its states due to asynchrony in agents' computations. As a result, we will in general have $x^i(k) \neq x^j(k)$ at all times k. We define the set K^i to contain all times k at which agent i updates x_i^i; agent i does not compute an update for time indices $k \notin K_i$. In designing an update law, we must provide robustness to asynchrony while allowing computations to be performed in a distributed fashion. First-order gradient descent methods are robust to many disturbances, and we therefore propose the following update law:

$$x_i^i(k+1) = \begin{cases} x_i^i(k) - \gamma_i \left(Q^{[i]} x^i(k) + r^{[i]} \right) & k \in K^i \\ x_i^i(k) & k \notin K^i \end{cases},$$

where agent i uses some stepsize $\gamma_i > 0$. This is equivalent to agent i using using the gradient descent law $x_i^i(k+1) = x_i^i(k) - \gamma_i \nabla_i f \left(x^i(k) \right)$ when it updates. The advantage of the block-based update law can be seen above, as agent i only needs to know $Q^{[i]}$ and $r^{[i]}$. Requiring each agent to store the entirety of Q and r would require $O(n^2)$ storage space, while $Q^{[i]}$ and $r^{[i]}$ only require $O(n)$. For large quadratic programs, this block-based update law dramatically reduces each agent's onboard storage requirements.

In order to account for communication delays, we use $\tau_j^i(k)$ to denote the time at which the value of $x_j^i(k)$ was originally computed by agent j. For example, if agent j computes a state update at time k_a and immediately transmits it to agent i, then agent i may receive this state update at time $k_b > k_a$ due to communication delays. Then τ_j^i is defined so that $\tau_j^i(k_b) = k_a$, which relates the time of receipt by agent i to the time at which agent j originally computed the piece of data being sent. For K^i and τ_j^i, we assume the following.

Assumption 3: For all $i \in [N]$, the set K^i is infinite. Moreover, for all $i \in [N]$ and $j \in [N] \backslash \{i\}$, if $\{k_d\}_{d \in \mathbb{N}}$ is a sequence in K^i tending to infinity, then

$$\lim_{d \to \infty} \tau_j^i(k_d) = \infty. \qquad \qquad \triangle$$

Assumption 3 is simply a formalization of the requirement that no agent ever permanently stop updating and sharing its own state with any other agent. For $i \neq j$, the sets K^i and K^j do not need to have any relationship because agents' updates are asynchronous. Our proposed update law for all agents can then be written as follows.

Algorithm 1: For all $i \in [N]$ and $j \in [N] \backslash \{i\}$, execute

$$x_i^i(k+1) = \begin{cases} x_i^i(k) - \gamma_i \left(Q^{[i]} x^i(k) + r^{[i]} \right) & k \in K^i \\ x_i^i(k) & k \notin K^i \end{cases}$$

$$x_j^i(k+1) = \begin{cases} x_j^j \left(\tau_j^i(k+1) \right) & \text{i receives j's state at k+1} \\ x_j^i(k) & \text{otherwise} \end{cases} \qquad \diamond$$

In Algorithm 1 we see that x_j^i changes only when agent i receives a transmission directly from agent j; otherwise it remains constant. This implies that agent i can update its own state using an old value of agent j's state multiple times and can reuse different agents' states different numbers of times. Showing convergence of this update law

must account for these delays, in addition to providing stepsize bounds for each agent, and that is the subject of the next section.

IV. CONVERGENCE OF ASYNCHRONOUS OPTIMIZATION

In this section, we prove convergence of the multi-agent block update law in Algorithm 1. This will be shown using a block-maximum norm to measure convergence, along with a collection of nested sets to show Lyapunov-like convergence. We will derive stepsize bounds from these concepts that will be used to show asymptotic convergence of all agents.

A. Block-Maximum Norms

The convergence of Algorithm 1 will be measured using a block-maximum norm as in [25], [14], and [24]. We do this to permit agents to use independent normalizations in Problem 1 to weight different components of x differently when estimating convergence to an optimum. Below, we refer to x_i^i as the ith block of x^i and x_j^i as the jth block of x^i. We next define the block-maximum norm that will be used to measure convergence.

Definition 1: Let $x \in \mathbb{R}^n$ consist of N blocks, with $x_i \in \mathbb{R}^{n_i}$ being the ith block. The ith block is weighted by some normalization constant $\omega_i \geq 1$ and is measured in the p_i-norm for some $p_i \in [1, \infty]$. The norm of the full vector x is defined as the maximum norm of any single block, i.e.,

$$\|x\|_{max} := \max_{i \in [N]} \frac{\|x_i\|_{p_i}}{\omega_i}. \qquad \blacktriangle$$

The following lemma allows us to upper-bound the induced block-maximum matrix norm by the Euclidean matrix norm, which will be used below in our convergence analysis.

Lemma 1: Suppose for all $i \in [N]$ that agent i uses the weight $\omega_i \geq 1$ and p_i-norm, $p_i \in [1, \infty]$, in the above block-maximum norm. Let $p_{min} := \min_{i \in [N]} p_i$ and let $\omega_{min} := \min_{i \in [N]} \omega_i$. Then for all $B \in \mathbb{R}^{n \times n}$,

$$\|B\|_{max} \leq \begin{cases} n^{\left(p_{min}^{-1} - \frac{1}{2}\right)} \omega_{min}^{-1} \|B\|_2 & p_{min} < 2 \\ \omega_{min}^{-1} \|B\|_2 & p_{min} \geq 2 \end{cases}.$$

Proof: See Lemma 1 in [26]. ∎

B. Convergence Via Lyapunov Sub-Level Sets

We will now analyze the convergence of Algorithm 1 when agents are communicating asynchronously. In order to show convergence, we construct a sequence of sets, $\{X(s)\}_{s \in \mathbb{N}}$, based on work in [25] and [14]. These sets behave analogously to sub-level sets of a Lyapunov function, and they will enable an invariance type argument in our convergence proof. Below, we use $\hat{x} := \arg\min_{x \in X} f(x)$ for the minimizer of f. For simplicity, we assume that $\hat{x}$ is in the interior of X, though all of our results hold without modification if a projection onto X_i is added to agent i's update in Algorithm 1, and that is the only change required if $\hat{x}$ is not in the interior of X. We state the following assumption on these sets, and below we will construct a sequence of sets that satisfies this assumption.

Assumption 4: There exists a collection of sets $\{X(s)\}_{s \in \mathbb{N}}$ that satisfies:

1) $... \subset X(s+1) \subset X(s) \subset ... \subset X$

2) $\lim_{s \to \infty} X(s) = \{\hat{x}\}$

3) There exists $X_i(s) \subset X_i$ for all $i \in [N]$ and $s \in \mathbb{N}$ such that $X(s) = X_1(s) \times ... \times X_N(s)$

4) $\theta_i(y) \in X_i(s+1)$, where $\theta_i(y) := y_i - \gamma_i \nabla_i f(y)$ for all $y \in X(s)$ and $i \in [N]$. $\triangle$

Assumptions 4.1 and 4.2 jointly guarantee that the collection $\{X(s)\}_{s \in \mathbb{N}}$ is nested and that they converge to a singleton containing $\hat{x}$. Assumption 4.3 allows for the blocks of x to be updated independently by the agents, which allows for decoupled update laws. Assumption 4.4 ensures that state updates make only forward progress toward $\hat{x}$, which ensures that each set is forward-invariant in time. It is shown in [25] and [14] that the existence of such a sequence of sets implies asymptotic convergence of the asynchronous update law in Algorithm 1. We therefore use this strategy to show asymptotic convergence in this paper. We propose to use the construction

$$X(s) = \{y \in X : \|y - \hat{x}\|_{max} \leq q^s n D_o\}, \tag{1}$$

where we define $D_o := \max_{i \in [N]} \|x^i(0) - \hat{x}\|_{max}$, which is the block furthest from $\hat{x}$ onboard any agent at timestep zero, and where we define the constant

$$q = \|I - \Gamma Q\|_2,$$

with $\Gamma = \text{diag}(\gamma_1 I_{n_1}, ..., \gamma_N I_{n_N})$. We will use the fact that each update contracts towards $\hat{x}$ by a factor of q, and the following theorem will establish bounds on every γ_i that imply $q \in (0, 1)$. This result will be used to show convergence of Algorithm 1 through satisfaction of Assumption 4.

Theorem 1: Let $Q = Q^T \succ 0$, $Q \in \mathbb{R}^{n \times n}$, have condition number k_Q, and let $\Gamma = \text{diag}(\gamma_1 I_{n_1}, ..., \gamma_N I_{n_N})$. If

$$\gamma_i \in \left(\frac{\sqrt{k_Q} - 1}{\|Q\|_2 \sqrt{k_Q}}, \frac{\sqrt{k_Q} + 1}{\|Q\|_2 \sqrt{k_Q}} \right) \quad \text{for all} \quad i \in [N], \tag{2}$$

then $\|I - \Gamma Q\|_2 < 1$.

Proof: To avoid interrupting the flow of the paper, proof of Theorem 1 can be found in the appendix. ∎

In Theorem 1, we note that any choice of γ_{lower} and γ_{upper} that satisfies

$$\frac{1}{2\gamma_{upper} \|Q\|_2} \left(\frac{(\sqrt{k_Q} + 1)^2}{\sqrt{k_Q}} - \frac{\gamma_{upper}}{\gamma_{lower}} \frac{(\sqrt{k_Q} - 1)^2}{\sqrt{k_Q}} \right) > 1 \tag{3}$$

defines a valid interval of step sizes to ensure $\|I - \Gamma Q\|_2 < 1$. While the algebra is omitted here, it can be shown that the range defined in Theorem 1 is the largest of these intervals. Note also that due to the structure of Equation (3), if the exact values of k_Q and $\|Q\|_2$ are unavailable or difficult to compute, then they can be replaced in Equation (2) by upper bounds. These could include using Gershgorin's Circle Theorem or the trace of Q to bound $\|Q\|_2$ and using results such as in [27] to bound k_Q.

Letting $\gamma_i \in \left(\frac{\sqrt{k_Q} - 1}{\|Q\|_2 \sqrt{k_Q}}, \frac{\sqrt{k_Q} + 1}{\|Q\|_2 \sqrt{k_Q}} \right)$ for all $i \in [N]$ and recalling our construction of sets $\{X(s)\}_{s \in \mathbb{N}}$ as

$$X(s) = \{y \in X : \|y - \hat{x}\|_{max} \leq q^s n D_o\},$$

we next show that Assumption 4 is satisfied, thereby ensuring convergence of Algorithm 1.

Theorem 2: If Γ satisfies the conditions in Theorem 1, then the collection of sets $\{X(s)\}_{s\in\mathbb{N}}$ as defined in Equation (1) satisfies Assumption 4.

Proof: For Assumption 4.1, by definition we have

$$X(s+1) = \left\{y \in X : \|y - \hat{x}\|_{max} \le q^{s+1}nD_o\right\}.$$

Since $q \in (0,1)$, we have $q^{s+1} < q^s$, which results in $\|y - \hat{x}\|_{max} \le q^{s+1}nD_o < q^snD_o$. Then $y \in X(s+1)$ implies $y \in X(s)$ and $X(s+1) \subset X(s) \subset X$, as desired.

For Assumption 4.2 we find

$$\lim_{s\to\infty} X(s) = \lim_{s\to\infty} \left\{y \in X : \|y - \hat{x}\|_{max} \le q^snD_o\right\} = \{\hat{x}\}.$$

The structure of the weighted block-maximum norm then allows us to see that $\|y - \hat{x}\|_{max} \le q^snD_o$ if and only if $\frac{1}{\omega_i}\|y_i - \hat{x}_i\|_{p_i} \le q^snD_o$ for all $i \in [N]$. It then follows that

$$X_i(s) = \left\{y_i \in X_i : \frac{1}{\omega_i}\|y_i - \hat{x}_i\|_{p_i} \le q^snD_o\right\},$$

which gives $X(s) = X_1(s) \times ... \times X_N(s)$, thus satisfying Assumption 4.3.

We then see that, for $y \in X(s)$,

$$\frac{\|\theta_i(y) - \hat{x}_i\|_{p_i}}{\omega_i} = \frac{1}{\omega_i}\left\|y_i - \gamma_i\left(Q^{[i]}y + r^{[i]}\right) - \hat{x}_i + \gamma_i\left(Q^{[i]}\hat{x} + r^{[i]}\right)\right\|_{p_i},$$

which follows from the definition of $\theta_i(y)$ and the fact that $\nabla_i f(\hat{x}) = 0$. We then find

$$\frac{\|\theta_i(y) - \hat{x}_i\|_{p_i}}{\omega_i} \le \max_{i\in[N]} \frac{1}{\omega_i}\|y_i - \gamma_i\left(Q^{[i]}y + r^{[i]}\right) - \hat{x}_i + \gamma_i\left(Q^{[i]}\hat{x} + r^{[i]}\right)\|_{p_i}$$

$$= \|y - \hat{x} - \Gamma\left(Qy + r\right) + \Gamma\left(Q\hat{x} + r\right)\|_{max},$$

which follows from our definition of the block-maximum norm. Continuing, we find

$$\frac{\|\theta_i(y) - \hat{x}_i\|_{p_i}}{\omega_i} \le \|y - \hat{x} - \Gamma Q\left(y - \hat{x}\right)\|_{max}$$

$$\le \|I - \Gamma Q\|_{max}\|y - \hat{x}\|_{max}$$

$$\le \begin{cases} \frac{n^{\left(p_{min}^{-1} - \frac{1}{2}\right)}}{\omega_{min}}\|I - \Gamma Q\|_2\|y - \hat{x}\|_{max} & p_{min} < 2 \\ \frac{1}{\omega_{min}}\|I - \Gamma Q\|_2\|y - \hat{x}\|_{max} & p_{min} \ge 2 \end{cases},$$

where the last inequality follows from Lemma 1. Seeing that $\|I - \Gamma Q\|_2 = q \in (0,1)$, and using the hypothesis

that $y \in X(s)$, we find

$$\frac{\|\theta_i(y) - \hat{x}_i\|_{p_i}}{\omega_i} \leq \begin{cases} \frac{n^{\left(p_{min}^{-1} - \frac{1}{2}\right)}}{\omega_{min}} q\|y - \hat{x}\|_{max} & p_{min} < 2 \\ \frac{1}{\omega_{min}} q\|y - \hat{x}\|_{max} & p_{min} \geq 2 \end{cases}$$

$$\leq \begin{cases} \frac{n^{\left(p_{min}^{-1} - \frac{1}{2}\right)}}{\omega_{min}} q^{s+1} D_o & p_{min} < 2 \\ \frac{1}{\omega_{min}} q^{s+1} D_o & p_{min} \geq 2 \end{cases}$$

$$\leq \begin{cases} q^{s+1} n D_o & p_{min} < 2 \\ q^{s+1} n D_o & p_{min} \geq 2 \end{cases},$$

where the bottom case follows from $\omega_{min} \geq 1$ and the top case follows from $\omega_{min} \geq 1$ and $p_{min}^{-1} - \frac{1}{2} < 1$ (since $p_i \in [1, \infty]$ for all $i \in [N]$). Then $\theta_i(y) \in X_i(s+1)$ and Assumption 4.4 is satisfied. ∎

Regarding Problem 1, we therefore state the following:

Theorem 3: Algorithm 1 solves Problem 1 and asymptotically converges to $\hat{x}$.

Proof: Theorem 2 shows the construction of the sets $\{X(s)\}_{s \in \mathbb{N}}$ satisfies Assumption 4, and from [25] and [14] we see this implies convergence of Algorithm 1 for all $i \in [N]$. The total asynchrony required by Problem 1 is incorporated by not requiring delay bounds, and agents do not require any coordination in selecting stepsizes, which means that all of the criteria of Problem 1 are satisfied. ∎

From these requirements, we see that agent i only needs to be initialized with $Q^{[i]}$, $r^{[i]}$, and upper bounds on $\|Q\|_2$ and k_Q, which can be set by a network operator. Agents are then free to choose normalizations and stepsizes independently, provided stepsizes obey the bounds established in Theorem 1.

V. INDEPENDENTLY REGULARIZED QUADRATIC PROGRAMS

Equation (2) quantifies the relationship between the condition number of Q and agents' stepsize bounds. For a perfectly conditioned matrix Q, such as the identity, we have $k_Q = 1$, which implies that stepsizes may be chosen from the open interval $\left(0, \frac{2}{\|Q\|_2}\right)$. This is the familiar $\frac{2}{L}$ bound encountered in conventional gradient descent settings. As the problem becomes more ill-conditioned and k_Q increases, the allowable interval of stepsize choices becomes narrower in Equation (2). Specifically, as $k_Q \to \infty$, the set of allowable stepsizes approaches the degenerate interval $\left[\frac{1}{\|Q\|_2}, \frac{1}{\|Q\|_2}\right]$, implying that all stepsizes must be equal in such a case.

Equation (2) suggests that if the condition number of a quadratic program is reduced, then the interval of allowable step sizes can be lengthened. In particular, regularizing f can improve k_Q to do so. As stated in Problem 2, we want to allow agents to choose regularization parameters independently. Before we analyze the effects of independently chosen regularizations on k_Q, we must first show that an algorithm that utilizes them will preserve the convergence properties of Algorithm 1. As shown above, a regularized cost function takes the form

$$f_A(x) := \frac{1}{2} x^T (Q + A)x + r^T x,$$

where $Q + A$ is symmetric positive definite because $Q = Q^T \succ 0$. We now state the following theorem that confirms that minimizing f_A succeeds.

Theorem 4: Suppose that $A \succ 0$ is diagonal with positive diagonal entries. Then Algorithm 1 satisfies the conditions stated in Problem 2 when f_A is minimized.

Proof: Replacing Q with $Q + A$, all assumptions and conditions used to prove Theorem 3 hold, with the only modifications being that the network will converge to $\hat{x}_A := \arg\min_{x \in X} f_A(x)$, and the agents must now be initialized with $Q^{[i]}$, $r^{[i]}$, and upper limits on $\|Q + A\|_2$ and the condition number of $Q + A$. ∎

Theorem 4 implies that there must be some known upper bound on $\|Q + A\|_2$ and the condition number of $Q + A$. These bounds can be determined using bounds on agents' allowable regularization parameters, which we develop now. First we will establish the following theorem, which demonstrates that independent regularizations can indeed reduce the condition number of the quadratic program.

Theorem 5: Let there be two $n \times n$ matrices $Q = Q^T \succ 0$ and $A = \text{diag}\,(\alpha_1 I_{n_1}, ..., \alpha_N I_{n_N}) \succ 0$ with respective condition numbers $k_Q = \frac{\lambda_1(Q)}{\lambda_n(Q)}$ and $k_A = \frac{\lambda_1(A)}{\lambda_n(A)} = \frac{\alpha_{max}}{\alpha_{min}}$, where $\alpha_{max} = \max_i \alpha_i$ and $\alpha_{min} = \min_i \alpha_i$. If $\frac{\alpha_{max}}{\alpha_{min}} < k_Q$, then $k_{Q+A} < k_Q$.

Proof: Using $\frac{\alpha_{max}}{\alpha_{min}} = k_A < k_Q$, we find $\frac{\lambda_1(A)}{\lambda_n(A)} < \frac{\lambda_1(Q)}{\lambda_n(Q)}$. Rearranging, we find $\lambda_n(Q)\lambda_1(A) < \lambda_1(Q)\lambda_n(A)$. Adding $\lambda_1(Q)\lambda_n(Q)$ to both sides and factoring gives

$$\lambda_n(Q)\left(\lambda_1(Q) + \lambda_1(A)\right) < \lambda_1(Q)\left(\lambda_n(Q) + \lambda_n(A)\right),$$

which we rearrange again to find

$$\frac{\lambda_1(Q) + \lambda_1(A)}{\lambda_n(Q) + \lambda_n(A)} < \frac{\lambda_1(Q)}{\lambda_n(Q)}. \tag{4}$$

From Weyl's inequalities, if B and C are $n \times n$ Hermitian matrices, then $\lambda_1(B + C) \leq \lambda_1(B) + \lambda_1(C)$ and $\lambda_n(B) + \lambda_n(C) \leq \lambda_n(B + C)$ [28, Fact 5.12.2]. Therefore

$$\frac{\lambda_1(Q + A)}{\lambda_n(Q + A)} \leq \frac{\lambda_1(Q) + \lambda_1(A)}{\lambda_n(Q) + \lambda_n(A)}.$$

Combining this with Equation (4) completes the proof. ∎

Since the regularization matrix A is chosen by the agents, it is always possible to have $k_A < k_Q$ provided $k_Q > 1$. Of course, we will only regularize such a problem, and thus, for all practical purposes, regularizing improves the conditioning of our problems as long as k_A is better-conditioned than k_Q. If we wish to reduce k_{Q+A} such that it is bounded above by some desired condition number k_D, we can bound α_{min} via

$$\alpha_{min} > \|Q\|_2 \left(k_D^{-1} - k_Q^{-1}\right) + \frac{\epsilon \|Q\|_2^2}{k_Q k_D \left(\|r\|_2 k_Q - \epsilon \|Q\|_2\right)}, \tag{5}$$

which works for any k_D such that

$$k_D > k_Q - \frac{\epsilon \|Q\|_2 \left(k_Q - 1\right)}{\|r\|_2 k_Q}.$$

Theorem 5 suggests that if we want to expand the interval of allowable stepsizes via regularizing, then we should choose large, homogeneous regularization parameters among agents. However, larger regularizations will lead to larger errors in the solution to a quadratic program, and the ability to take larger steps must be balanced with the quality of solution obtained. In the next section, we will quantify the relationship between regularizations and

error, and develop a set of guidelines to govern agents' selection of regularization parameters based on desired error bounds.

VI. REGULARIZATION ERROR BOUND

We see from the structure of the quadratic program that the solution to the unregularized minimization problem is $\hat{x} = -Q^{-1}r$, where Q^{-1} is well-defined because $Q \succ 0$. Similarly, the solution to the regularized minimization problem is $\hat{x}_A = -(Q + A)^{-1}r$. We therefore define the regularization error when using regularization matrix A as $e_A := \|\hat{x} - \hat{x}_A\|_2$. We now upper bound e_A in terms of the entries of A.

Theorem 6: For the agent-specified regularization matrix $A = \text{diag}\,(\alpha_1 I_{n_1}, ..., \alpha_N I_{n_N})$, the regularization error $e_A := \|\hat{x} - \hat{x}_A\|_2$ is bounded via

$$e_A \leq \frac{\|r\|_2 k_Q^2 \alpha_{max}}{\|Q\|_2^2 + \|Q\|_2 k_Q \alpha_{max}}, \tag{6}$$

where $\alpha_{max} := \max_i \alpha_i$.

Proof: By definition, we have

$$e_A = \|Q^{-1}r - (Q + A)^{-1}r\|_2 \leq \|Q^{-1} - (Q + A)^{-1}\|_2 \|r\|_2.$$

The above inequality can be rewritten using Lemma 2.

Lemma 2: Let Q and A be positive definite matrices. Then

$$Q^{-1} - (Q + A)^{-1} = (QA^{-1}Q + Q)^{-1}.$$

Proof: From the Woodbury matrix identity,

$$(B + UCV)^{-1} = B^{-1} - B^{-1}U(C^{-1} + VB^{-1}U)^{-1}VB^{-1}.$$

Let $B = U = C = Q$ and $C = Q^{-1}AQ^{-1}$. Then

$$(Q + A)^{-1} = Q^{-1} - Q^{-1}Q(QA^{-1}Q + QQ^{-1}Q)^{-1}QQ^{-1},$$

which simplifies to $(Q + A)^{-1} = Q^{-1} - (QA^{-1}Q + Q)^{-1}$. Rearranging completes the lemma. ∎

Therefore, $e_A \leq \|(QA^{-1}Q + Q)^{-1}\|_2 \|r\|_2$ using Lemma 2. Note that $QA^{-1}Q + Q$ is a symmetric positive definite matrix. For any symmetric positive definite matrix M, we know $\|M^{-1}\|_2 = \lambda_1(M^{-1}) = \lambda_n^{-1}(M)$. Applying this to $(QA^{-1}Q + Q)^{-1}$ gives

$$e_A \leq \frac{\|r\|_2}{\lambda_n(QA^{-1}Q + Q)}.$$

From Weyl's Inequalities [28, Fact 5.12.2] we then have

$$e_A \leq \frac{\|r\|_2}{\lambda_n(QA^{-1}Q) + \lambda_n(Q)}.$$

Since $QA^{-1}Q$ is similar to Q^2A^{-1}, we know that $\lambda_n(QA^{-1}Q) = \lambda_n(Q^2A^{-1})$, and since Q^2 and A^{-1} are both symmetric positive definite, and $\lambda_n(Q^2) = \lambda_n^2(Q)$, we can say [28, Fact 8.19.18] that $\lambda_n^2(Q)\lambda_n(A^{-1}) \leq \lambda_n(Q^2A^{-1})$, which gives

$$e_A \leq \frac{\|r\|_2}{\lambda_n^2(Q)\lambda_n(A^{-1}) + \lambda_n(Q)} = \frac{\|r\|_2}{\lambda_n^2(Q)\alpha_{max}^{-1} + \lambda_n(Q)} = \frac{\|r\|_2 \alpha_{max}}{\lambda_n^2(Q) + \lambda_n(Q)\alpha_{max}}.$$

Substituting in $\lambda_n(Q) = \frac{\|Q\|_2}{k_Q}$ completes the proof. ∎

VII. Discussion of Results

In this section, we briefly provide some remarks and interpretation of our results. We begin with the following corollary that follows from Theorem 6.

Corollary 1: As $\alpha_{max} \to 0$, $e_A \to 0$.

It is desirable for an error bound to be sharp in the sense that equality is achieved for at least one point. Corollary 1 shows that this is achieved in Equation (6). This implies zero regularization error when zero regularization is applied, which establishes Equation (6) as a sharp upper bound.

Corollary 2: As $\alpha_{max} \to \infty$, $e_A \to \frac{\|r\|_2 k_Q}{\|Q\|_2}$.

Another notable behavior of Equation (6) is a finite upper bound on the regularization error, as shown in Corollary 2. To elaborate on this, consider the case where $A = \alpha I$, where α is a positive scalar. As $\alpha \to \infty$, we see $\hat{x}_A = -(Q + A)^{-1}r \to 0$. That is, very large regularizations move the regularized solution $\hat{x}_A$ towards zero. We also see as $\alpha \to \infty$, $e_A \to \|Q^{-1}r\|_2$, since the norm of the error between zero and the unregularized solution is simply the norm of the unregularized solution itself. The upper bound from Equation (6) as $\alpha_{max} \to \infty$ is due to the fact that

$$\|Q^{-1}r\|_2 \leq \|Q^{-1}\|_2 \|r\|_2 = \frac{\|r\|_2 k_Q}{\|Q\|_2}.$$

If there is some desired upper bound ϵ on the error e_A, then an upper bound on α_{max} can be determined as

$$\alpha_{max} < \frac{\epsilon \|Q\|_2^2}{\|r\|_2 k_Q^2 - \epsilon \|Q\|_2 k_Q}, \tag{7}$$

which holds as long as $\epsilon < \frac{\|r\|_2 k_Q}{\|Q\|_2}$. This bound can be combined with the bound on α_{min} in Equation (5) to present agents with an admissible range of regularization parameters, which we do in the next section.

VIII. Simulation

In this section, two simulations are run in which a network of agents solves a quadratic program using Algorithm 1. In the first simulation, the problem is unregularized and agents select stepsizes independently from the allowable set for the problem, and they then asynchronously solve the quadratic program. In the second simulation, a desired condition number bound and regularization error bound are specified. Agents independently select regularization parameters to satisfy those requirements. Agents then choose stepsizes independently from the expanded allowable set, and then asynchronously solve the quadratic program. The desired condition number bound and error bound conditions are shown to be satisfied, and the convergence histories of the first and second simulations are compared.

The network consists of 25 agents, each with 4 states. The quadratic program is randomly generated with $k_Q = 100$ and $\|Q\|_2 = 100$. For the first simulation, agents randomly choose stepsizes from the interval $(0.009, 0.0110)$, which is obtained from Equation (2) and the values of k_Q and $\|Q\|_2$. In the second, we specify the desired condition number $k_D = 10$ and the upper bound $\epsilon = 0.1$ on regularization error. Agents then randomly choose regularization
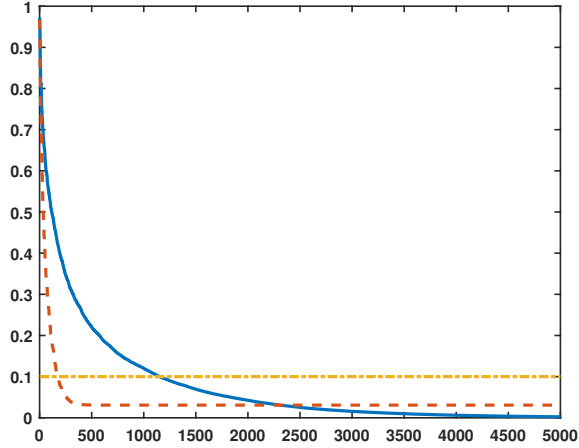
Fig. 1. The distance to the optimum $\hat{x}$ for the unregularized problem (shown in the solid blue line) and the regularized problem (shown in the dashed orange line). The desired regularization error bound ϵ is shown as the dash-dotted yellow line. We see that the regularized problem converges substantially faster than the unregularized one. Regularization error is equal to the final value of the dashed orange line, which we see is indeed below the dash-dotted yellow line, indicating that the desired regularization error is indeed obeyed.

parameters from the interval $(11, 20)$, which follows from Equations (7) and (5). Agents then randomly select stepsizes from the interval $(0.0056, 0.0117)$, which is obtained as before.

Both simulations were run for 2000 timesteps. To force asynchrony, at each timestep agent i has a 10% chance of transmitting its state to agent j, and agent i has a 10% chance of computing an update to its own state.

As shown in Figure 1, the regularized problem (dashed orange line) shows notably faster convergence than the unregularized one (solid blue line). The final error in the regularized solution was $e_A = 0.0308$, well below the desired upper bound of $e_A < 0.1$, which is shown in Figure 1 by the dash-dotted yellow line. The condition number of the regularized problem was $k_{Q+A} = 8.1836$, satisfying our desired condition number bound as well.

IX. CONCLUSION

A totally asynchronous quadratic programming framework was presented. This framework allowed agents to independently choose all stepsize and regularization parameters and showed fast convergence in simulation. Future work includes incorporating functional constraints into this framework, as well as implementing this work for path planning problems in robotic teams.

APPENDIX

Proof of Theorem 1: The following lemmas will facilitate the proof of Theorem 1.

Lemma 3: Let B and C be positive definite $n \times n$ Hermitian matrices with eigenvalues $\lambda_1(B) \geq \lambda_2(B) \geq ... \geq \lambda_n(B)$ and $\lambda_1(C) \geq \lambda_2(C) \geq ... \geq \lambda_n(C)$. Then

$$\lambda_n(BC + CB) \geq \min_{\beta \in \{1,n\}} \left[\frac{\lambda_\beta(B)\lambda_\beta(C)}{2} \left(\frac{(\sqrt{k_B}+1)^2}{\sqrt{k_B}} - k_C \frac{(\sqrt{k_B}-1)^2}{\sqrt{k_B}} \right) \right],$$

where k_B and k_C are the spectral condition numbers of B and C, respectively.

Proof: See [29]. ∎

Lemma 4: Let $Q = Q^T \succ 0$, $Q \in \mathbb{R}^{n \times n}$ have condition number k_Q, and let $\Gamma = \text{diag}\left(\gamma_1 I_{n_1}, ..., \gamma_N I_{n_N}\right)$. If

$$\gamma_i \in \left(\frac{\sqrt{k_Q} - 1}{\|Q\|_2 \sqrt{k_Q}}, \frac{\sqrt{k_Q} + 1}{\|Q\|_2 \sqrt{k_Q}} \right) \text{ for all } i \in [N],$$

then $Q^{-1}\Gamma^{-1} + \Gamma^{-1}Q^{-1} - I \succ 0$.

Proof: Let $B = Q^{-1}$ and $C = \Gamma^{-1}$, and note that

$$k_{Q^{-1}} = \frac{\lambda_1(Q^{-1})}{\lambda_n(Q^{-1})} = \frac{\lambda_n^{-1}(Q)}{\lambda_1^{-1}(Q)} = \frac{\lambda_1(Q)}{\lambda_n(Q)} = k_Q$$

and likewise $k_{\Gamma^{-1}} = k_\Gamma$. Now, using Lemma 3, we can write

$$\lambda_n(Q^{-1}\Gamma^{-1} + \Gamma^{-1}Q^{-1}) \geq \min_{\beta \in \{1, n\}} \left[\frac{\lambda_\beta^{-1}(Q)\lambda_\beta^{-1}(\Gamma)}{2} \left(\frac{(\sqrt{k_Q} + 1)^2}{\sqrt{k_Q}} - k_\Gamma \frac{(\sqrt{k_Q} - 1)^2}{\sqrt{k_Q}} \right) \right]. \qquad (8)$$

Define the constants

$$\gamma_{lower} = \frac{\sqrt{k_Q} - 1}{\|Q\|_2 \sqrt{k_Q}} \text{ and } \gamma_{upper} = \frac{\sqrt{k_Q} + 1}{\|Q\|_2 \sqrt{k_Q}}.$$

Then, by hypothesis, $\gamma_i \in (\gamma_{lower}, \gamma_{upper})$ and

$$k_\Gamma < \frac{\gamma_{upper}}{\gamma_{lower}} = \frac{\sqrt{k_Q} + 1}{\sqrt{k_Q} - 1}.$$

Substituting this bound into Equation (8), we find

$$\lambda_n(Q^{-1}\Gamma^{-1} + \Gamma^{-1}Q^{-1}) \geq \min_{\beta \in \{1,n\}} \left[\frac{\lambda_\beta^{-1}(Q)\lambda_\beta^{-1}(\Gamma)}{2} \left(\frac{(\sqrt{k_Q} + 1)^2}{\sqrt{k_Q}} - \frac{(\sqrt{k_Q} + 1)}{(\sqrt{k_Q} - 1)} \frac{(\sqrt{k_Q} - 1)^2}{\sqrt{k_Q}} \right) \right]$$

$$\geq \min_{\beta \in \{1,n\}} \left[\frac{\lambda_\beta^{-1}(Q)\lambda_\beta^{-1}(\Gamma)}{2} \left(2 \frac{\sqrt{k_Q} + 1}{\sqrt{k_Q}} \right) \right],$$

the right hand side of which is always positive. This indicates that the minimum will occur when $\beta = 1$, and gives

$$\lambda_n(Q^{-1}\Gamma^{-1} + \Gamma^{-1}Q^{-1}) \geq \lambda_1^{-1}(Q)\lambda_1^{-1}(\Gamma) \left(\frac{\sqrt{k_Q} + 1}{\sqrt{k_Q}} \right).$$

Using $\lambda_1(Q) = \|Q\|_2$ and $\lambda_1(\Gamma) < \gamma_{upper}$, we have

$$\lambda_n(Q^{-1}\Gamma^{-1} + \Gamma^{-1}Q^{-1}) > \left(\frac{1}{\|Q\|_2} \right) \left(\frac{\|Q\|_2 \sqrt{k_Q}}{\sqrt{k_Q} + 1} \right) \left(\frac{\sqrt{k_Q} + 1}{\sqrt{k_Q}} \right)$$

$$= 1,$$

which implies $Q^{-1}\Gamma^{-1} + \Gamma^{-1}Q^{-1} \succ I$. ∎

From $Q^{-1}\Gamma^{-1} + \Gamma^{-1}Q^{-1} - I \succ 0$ we see that the matrix on the left hand side is symmetric positive definite. In particular, all eigenvalues are positive. Then, by Sylvester's Law of Inertia [28, Fact 5.8.17], performing a congruence transformation will preserve the positivity of the eigenvalues. Any non-singular matrix P provides a

valid congruence transformation of a matrix M via $M' = P^T M P$, where M' is the transformed matrix. Using the nonsingular matrix $P = \Gamma Q$, we find

$$P^T(Q^{-1}\Gamma^{-1} + \Gamma^{-1}Q^{-1} - I)P \succ 0$$

$$Q\Gamma(Q^{-1}\Gamma^{-1} + \Gamma^{-1}Q^{-1} - I)\Gamma Q \succ 0,$$

where we have used the symmetry of Q and Γ. Expanding, we find $Q\Gamma + \Gamma Q - Q\Gamma^2 Q \succ 0$. Multiplying by -1 and adding I gives $I - Q\Gamma - \Gamma Q + Q\Gamma^2 Q \prec I$, which we factor as $(I - \Gamma Q)^T(I - \Gamma Q) \prec I$. This in turn implies $\lambda_{max}[(I - \Gamma Q)^T(I - \Gamma Q)] < 1$, where the square root gives $\sqrt{\lambda_{max}[(I - \Gamma Q)^T(I - \Gamma Q)]} < 1$ and finally $\|I - \Gamma Q\|_2 < 1$. ∎

REFERENCES

[1] Z.-Q. Luo and W. Yu, "An introduction to convex optimization for communications and signal processing," *IEEE Journal on selected areas in communications*, vol. 24, no. 8, pp. 1426–1438, 2006.

[2] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.

[3] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 2520–2525.

[4] M. Chiang *et al.*, "Geometric programming for communication systems," *Foundations and Trends® in Communications and Information Theory*, vol. 2, no. 1–2, pp. 1–154, 2005.

[5] S. Shalev-Shwartz *et al.*, "Online learning and online convex optimization," *Foundations and Trends® in Machine Learning*, vol. 4, no. 2, pp. 107–194, 2012.

[6] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[7] A. I. Chen and A. Ozdaglar, "A fast distributed proximal-gradient method," in *2012 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2012, pp. 601–608.

[8] 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.

[9] A. Nedic, 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, 2010.

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

[11] S. S. Ram, A. Nedić, and V. V. Veeravalli, "Distributed stochastic subgradient projection algorithms for convex optimization," *Journal of optimization theory and applications*, vol. 147, no. 3, pp. 516–545, 2010.

[12] I. Lobel and A. Ozdaglar, "Distributed subgradient methods for convex optimization over random networks," *IEEE Transactions on Automatic Control*, vol. 56, no. 6, pp. 1291–1306, 2011.

[13] I. Rodriguez-Lujan, R. Huerta, C. Elkan, and C. S. Cruz, "Quadratic programming feature selection," *Journal of Machine Learning Research*, vol. 11, no. Apr, pp. 1491–1516, 2010.

[14] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation: numerical methods*. Prentice hall Englewood Cliffs, NJ, 1989, vol. 23.

[15] R. Carli, G. Notarstefano, L. Schenato, and D. Varagnolo, "Distributed quadratic programming under asynchronous and lossy communications via newton-raphson consensus," in *2015 European Control Conference (ECC)*. IEEE, 2015, pp. 2514–2520.

[16] A. Teixeira, E. Ghadimi, I. Shames, H. Sandberg, and M. Johansson, "Optimal scaling of the admm algorithm for distributed quadratic programming," in *52nd IEEE Conference on Decision and Control*. IEEE, 2013, pp. 6868–6873.

[17] C.-P. Lee and D. Roth, "Distributed box-constrained quadratic optimization for dual linear svm," in *International Conference on Machine Learning*, 2015, pp. 987–996.

[18] K. Lee and R. Bhattacharya, "On the convergence analysis of asynchronous distributed quadratic programming via dual decomposition," *arXiv preprint arXiv:1506.05485*, 2015.

[19] A. Kozma, J. V. Frasch, and M. Diehl, "A distributed method for convex quadratic programming problems arising in optimal control of distributed systems," in *52nd IEEE Conference on Decision and Control*. IEEE, 2013, pp. 1526–1531.

[20] M. Todescato, G. Cavraro, R. Carli, and L. Schenato, "A robust block-jacobi algorithm for quadratic programming under lossy communications," *IFAC-PapersOnLine*, vol. 48, no. 22, pp. 126–131, 2015.

[21] W. Ren, R. W. Beard, and E. M. Atkins, "A survey of consensus problems in multi-agent coordination," in *Proceedings of the 2005, American Control Conference, 2005*. IEEE, 2005, pp. 1859–1864.

[22] J. Koshal, A. Nedić, and U. V. Shanbhag, "Multiuser optimization: Distributed algorithms and error analysis," *SIAM Journal on Optimization*, vol. 21, no. 3, pp. 1046–1081, 2011.

[23] M. T. Hale, A. Nedić, and M. Egerstedt, "Cloud-based centralized/decentralized multi-agent optimization with communication delays," in *2015 54th IEEE Conference on Decision and Control (CDC)*, Dec 2015, pp. 700–705.

[24] M. T. Hale, A. Nedić, and M. Egerstedt, "Asynchronous multiagent primal-dual optimization," *IEEE Transactions on Automatic Control*, vol. 62, no. 9, pp. 4421–4435, 2017.

[25] D. P. Bertsekas and J. N. Tsitsiklis, "Convergence rate and termination of asynchronous iterative algorithms," in *Proceedings of the 3rd International Conference on Supercomputing*, 1989, pp. 461–470.

[26] S. Hochhaus and M. T. Hale, "Asynchronous distributed optimization with heterogeneous regularizations and normalizations," in *2018 IEEE Conference on Decision and Control (CDC)*, 2018, pp. 4232–4237.

[27] G. Cheng, "Note on some upper bounds for the condition number," *Journal of Mathematical Inequalities*, vol. 8, no. 2, pp. 369–374, 2014.

[28] D. S. Bernstein, *Matrix Mathematics: Theory, Facts, and Formulas-Second Edition*. Princeton university press, 2009.

[29] D. W. Nicholson, "Eigenvalue bounds for ab+ ba, with a, b positive definite matrices," *Linear Algebra and its Applications*, vol. 24, pp. 173–184, 1979.