Networked and Distributed Control Systems

(SC42100)

Assignment 4

Student
Daniel Varela - 5321263

Responsible Instructors
Tamás Keviczky
Manuel Mazo
Gabriel Gleizer

June 29, 2021

Contents

Introduction

This assignment looks at the practical aspect of the implementation of several distributed algorithms.

The motivating setting for implementing all algorithms is the control and coordination of four aircraft. The main objective is the convergence of this platoon to the same state within a limited time span.

To this end, Section 1.1 sees the implementation of projected subgradient method, and 1.2 looks at the influence of step size in the effectiveness of this first method. Section 1.3 shows a version of Nesterov's method and compares it to the projected subgradient method in terms of rate of convergence. Next, in Section 1.4 changes strategy, implements a combined consensus/incremental subgradient approach and looks at the effect of the effect of consensus iterations and the step-size on the algorithm. Finally, 2 sees yet another different approach to the problem, by finding the consensus variable via ADMM.

All code related to this Assingment can be found in this Github Repository.

1 Problem 1

The problem focuses on the coordination of a multi-aircraft system, composed of four different vehicles which start from different initial conditions and have different dynamics.

$$x_i(t+1) = Ax_i(t) + Bu_i(t), \quad x_i(0) = x_{i,0}, \quad i = 1, \dots, 4, \quad t = 0, \dots, T_{\text{final}} - 1 \tag{1}$$

The main objective is to coordinate all four aircraft to an optimal common target state at T_{final}, while satisfying some input constraints at all times. To this end, the objective is to minimize the following quadratic function:

$$J = \sum_i \sum_t x_i(t)^\intercal x_i(t) + u_i(t)^\intercal u_i(t) \tag{2}$$

From this description, it's possible to define the problem and constraints. As previously stated, the objective is to minimize the cost function in 2 subject to constraints on the inputs and on the final states. This is easily formalized in the following problem:

$$\begin{aligned} \min_{x_i, u_i} \quad & \sum_i \sum_t x_i(t)^\intercal x_i(t) + u_i(t)^\intercal u_i(t) \\ \text{subject to} \quad & x_1(T_{\text{final}}) = x_2(T_{\text{final}}) = x_3(T_{\text{final}}) = x_4(T_{\text{final}}) = x_f \\ & |u_i(t)| \leq \frac{u_{\max}}{T_{\text{final}}} \end{aligned} \tag{3}$$

Focusing on a single agent i, the it's possible to formulate the following prediction model:

$$\mathbf{x_i} = Rx_i(0) + S\mathbf{u_i}, \quad \begin{cases} \mathbf{x_i} = [x_i(1), \dots x_i(T_{\text{final}})] \\ \mathbf{u_i} = [u_i(0), \dots u_i(T_{\text{final}} - 1)] \end{cases} \tag{4}$$

Note, in particular, that the final equality constraints can be written in terms of the decision vector $\mathbf{u_i}$:

$$\begin{aligned} x_i(T_{\text{final}}) =& \overbrace{A_i^{T_{\text{final}}} x_i(0)}^{b_{\text{eq}}^i} + \overbrace{\begin{bmatrix} A_i^{T_{\text{final}}-1}B_i & A_i^{T_{\text{final}}-2}B_i & \dots & A_iB_i & B_i \end{bmatrix}}^{A_{\text{eq}}^i} \mathbf{u_i} \\ =& A_{\text{eq}}^i \mathbf{u_i} + b_{\text{eq}}^i \end{aligned} \tag{5}$$

The inequality constraints can also be reorganized in matrix form. Defining $\bar{u} = \frac{u_{\max}}{T_{\text{final}}}$:

$$|u_i(t)| \leq \bar{u} \Leftrightarrow |\mathbf{u_i}| \leq \mathbf{1}\bar{u} \Leftrightarrow \overbrace{\begin{bmatrix} I & 0 \\ 0 & -I \end{bmatrix}}^{A_{\text{in}}} \mathbf{u_i} \leq \overbrace{\begin{bmatrix} \mathbf{1}\bar{u} \\ \mathbf{1}\bar{u} \end{bmatrix}}^{b_{\text{in}}} \Leftrightarrow A_{\text{in}}\mathbf{u_i} - b_{\text{in}} \leq 0 \tag{6}$$

The cost function can be re-written as:

$$\begin{aligned} J =& \sum_i \sum_t x_i(t)^\intercal x_i(t) + u_i(t)^\intercal u_i(t) \\ =& \sum_i \mathbf{x_i}^\intercal \mathbf{x_i} + \mathbf{u_i}^\intercal \mathbf{u_i} \\ =& \left(Rx_i(0) + S\mathbf{u_i}\right)^\intercal \left(Rx_i(0) + S\mathbf{u_i}\right) + \mathbf{u_i}^\intercal \mathbf{u_i} \\ =& \mathbf{u_i}^\intercal \underbrace{\left(S^\intercal S + I\right)}_{H_i} \mathbf{u_i} + \underbrace{2\left(S^\intercal R x_i(0)\right)^\intercal}_{h_i^\intercal} \mathbf{u_i} + x_i(0)^\intercal R^\intercal R x_i(0) \\ =& \mathbf{u_i}^\intercal H_i \mathbf{u_i} + h_i^\intercal \mathbf{u_i} + x_i(0)^\intercal R^\intercal R x_i(0) = V(\mathbf{u_i}) + x_i(0)^\intercal R^\intercal R x_i(0) \end{aligned} \tag{7}$$

Dropping the constant term in the cost function, problem 3 can be recast as:

$$\begin{aligned} \min_{\mathbf{u_i}} \quad & V(\mathbf{u_1}) + V(\mathbf{u_2}) + V(\mathbf{u_3}) + V(\mathbf{u_4}) \\ \text{subject to} \quad & \begin{array}{l|l} A_{\text{eq}}^1\mathbf{u_1} + b_{\text{eq}}^1 = A_{\text{eq}}^2\mathbf{u_2} + b_{\text{eq}}^2 & A_{\text{in}}\mathbf{u_1} - b_{\text{in}} \leq 0 \\ A_{\text{eq}}^2\mathbf{u_2} + b_{\text{eq}}^2 = A_{\text{eq}}^3\mathbf{u_3} + b_{\text{eq}}^3 & A_{\text{in}}\mathbf{u_2} - b_{\text{in}} \leq 0 \\ A_{\text{eq}}^3\mathbf{u_3} + b_{\text{eq}}^3 = A_{\text{eq}}^4\mathbf{u_4} + b_{\text{eq}}^4 & A_{\text{in}}\mathbf{u_3} - b_{\text{in}} \leq 0 \\ A_{\text{eq}}^4\mathbf{u_4} + b_{\text{eq}}^4 = A_{\text{eq}}^1\mathbf{u_1} + b_{\text{eq}}^1 & A_{\text{in}}\mathbf{u_4} - b_{\text{in}} \leq 0 \end{array} \end{aligned} \tag{8}$$

Note that the final equality constraint isn't really necessary to the definition of the problem. It's sufficient to have $(x_f^1 = x_f^2)$, $(x_f^2 = x_f^3)$ and $(x_f^3 = x_f^4)$. However, as will be shown later, adding the fourth constraint $(x_f^4 = x_f^1)$ dramatically improves the convergence rate of the algorithm.

1.1 Projected Subgradient Method

To implement a projected subgradient method with this problem the first step is to understand what the subgradient will be in this context.

Since the basis to solve the problem is dual decomposition, the subgradient updates will be done over the dual variables. To get a grasp on this problem, one can start from a generic formulation of the optimization problem subject to one equality constraint:

$$\begin{aligned} \min_{\theta} \quad & f(\theta) = \sum_i f^i(\theta^i) \\ \text{subject to} \quad & h(\theta) = \sum_i h^i(\theta^i) = 0 \end{aligned} \tag{9}$$

The dual formulation of the problem at the higher level will be to maximize $d(\lambda)$:

$$\max_{\lambda} \{d(\lambda)\} = \max_{\lambda} \left\{ \inf_{\theta} \left(f(\theta) + \lambda^{\intercal} h(\theta) \right) \right\} \tag{10}$$

Note that this function is always concave. This means the definition of subgradient isn't quite correct, since that applies only to convex functions. One option here is to work with the negative of $d(\lambda)$. The higher level problem becomes:

$$\min_{\lambda} \{-d(\lambda)\} = \min_{\lambda} \left\{ - \inf_{\theta} \left(f(\theta) + \lambda^{\intercal} h(\theta) \right) \right\} \tag{11}$$

This is now a convex problem, so it's possible to proceed with finding the subgradient of the dual variable. From the definition, g is a subgradient for $f(\theta)$ at θ_0 if $g \in \partial f(\theta_0)$.

$$\partial f(\theta_0) = \left\{ g \in \mathbb{R}^M \mid f(\theta) \geq f(\theta_0) + g^{\top} (\theta - \theta_0), \forall \theta \in \mathbb{R}^M \right\} \tag{12}$$

For a dual variable λ^*, let θ^* be the minimizer of the Lagrangian:

$$d(\lambda^*) = \inf_{\theta} \left(f(\theta) + (\lambda^*)^{\intercal} h(\theta) \right) = f(\theta^*) + (\lambda^*)^{\intercal} h(\theta^*) \tag{13}$$

Then we know that, for a generic feasible dual variable λ, the following holds:

$$\begin{aligned} -d(\lambda) &= - \inf_{\theta} \left(f(\theta) + \lambda^{\intercal} h(\theta) \right) \\ &\geq - \left[f(\theta^*) + \lambda^{\intercal} h(\theta^*) \right] = - \left[f(\theta^*) + (\lambda - \lambda^* + \lambda^*)^{\intercal} h(\theta^*) \right] \\ &= - \left[f(\theta^*) + (\lambda^*)^{\intercal} h(\theta^*) + (\lambda - \lambda^*)^{\intercal} h(\theta^*) \right] = -d(\lambda^*) - (\lambda - \lambda^*)^{\intercal} h(\theta^*) \\ &= - d(\lambda^*) - h(\theta^*)^{\intercal} (\lambda - \lambda^*) \end{aligned} \tag{14}$$

From the definition of a subgradient, $-h(\theta^*)$ is a subgradient of $-d(\lambda)$ at λ^*.

Since the problem is a minimization of the dual variables (which can assume any real value for equality constraints on the primal problem), the subgradient updates are defined as:

$$\lambda_{k+1} = \lambda_k - \alpha_k g \Leftrightarrow \lambda_{k+1} = \lambda_k + \alpha h(\theta^*) \tag{15}$$

If the step parameter α_k is square summable and the subgradient $-h(\theta^*)$ is bounded, then the method is guaranteed to converge.

This simple example on how to find a subgradient of a dual variable can easily be extended to larger problems. In general, the subgradient of a dual variable λ_j in the formulation presented above will be the value of the constraint $h_j(\theta)$ that dual variable is associated to, calculated at the optimal primal point θ^*.

Going back to the problem in 8, the initial step is to write the Lagrangian of this problem with relaxed constraints. Defining dual variables $\lambda^{(i,j)}$ for the coupling equality constraints, and μ^i for the inequality constraints, the Lagrangian becomes:

$$\begin{aligned} L(\mathbf{u_i}, \lambda, \mu) =& V(\mathbf{u_1}) + V(\mathbf{u_2}) + V(\mathbf{u_3}) + V(\mathbf{u_4}) + \\ & \lambda^{(1,2)} \left(A^1_{\text{eq}} \mathbf{u_1} + b^1_{\text{eq}} - A^2_{\text{eq}} \mathbf{u_2} - b^2_{\text{eq}} \right) + \lambda^{(2,3)} \left(A^2_{\text{eq}} \mathbf{u_2} + b^2_{\text{eq}} - A^3_{\text{eq}} \mathbf{u_3} - b^3_{\text{eq}} \right) + \\ & \lambda^{(3,4)} \left(A^3_{\text{eq}} \mathbf{u_3} + b^3_{\text{eq}} - A^4_{\text{eq}} \mathbf{u_4} - b^4_{\text{eq}} \right) + \lambda^{(4,1)} \left(A^4_{\text{eq}} \mathbf{u_4} + b^4_{\text{eq}} - A^1_{\text{eq}} \mathbf{u_1} - b^1_{\text{eq}} \right) + \\ & \mu^1 \left(A_{\text{in}} \mathbf{u_1} - b_{\text{in}} \right) + \mu^2 \left(A_{\text{in}} \mathbf{u_2} - b_{\text{in}} \right) + \mu^3 \left(A_{\text{in}} \mathbf{u_3} - b_{\text{in}} \right) + \mu^4 \left(A_{\text{in}} \mathbf{u_4} - b_{\text{in}} \right) \end{aligned} \tag{16}$$

The dual master problem is that of maximizing the dual variables within the respective constraints.

$$\sup_{\mu\geq 0,\lambda} d(\lambda,\mu) = \sup_{\mu\geq 0,\lambda} \inf_{\mathbf{u_i}} L(\mathbf{u_i},\lambda,\mu) \tag{17}$$

Note that the dual variable associated to the inequality constraints is itself constrained, $\mu^i \geq 0$. Additionally, it's possible to solve a version of this problem where only the equality constraints are relaxed, and where the inequality constraints are imposed in each sub-problem. This, however, complicated matters for implementing Nesterov's method in Section 1.3, so the choice was made to follow this formulation from the start.

The obvious implication of this choice is that feasibility is only guaranteed asymptotically. That is, all constraints on the problem were relaxed, and are only met asymptotically, meaning both inequality and equality constraints may be violated in early iterations of the process, before convergence.

This problem is easily separable into 4 symmetric individual problems in each of the variables $\mathbf{u_i}$ and corresponding constraints. The four dual lower level unconstrained problems will be:

$$\min_{\mathbf{u_i}} V(\mathbf{u_i}) + \lambda^{(i,j)}\left(A^i_{\text{eq}}\mathbf{u_i} + b^i_{\text{eq}} - A^j_{\text{eq}}\mathbf{u_j} + b^j_{\text{eq}}\right) + \mu^i\left(A_{\text{in}}\mathbf{u_i} - b_{\text{in}}\right), \quad \begin{cases} i = 1,2,3,4 \\ j = 1 + \text{mod}(i,4) \end{cases} \tag{18}$$

The advantage of lifting all constraints in the sub-problems is that, since the problems are unconstrained, they can simply be solved analytically. For each agent:

$$\begin{aligned}
\mathbf{u}_i^* =& \arg\min_{\mathbf{u_i}} V(\mathbf{u_i}) + \lambda^{(i,j)}\left(A^i_{\text{eq}}\mathbf{u_i} + b^i_{\text{eq}} - A^j_{\text{eq}}\mathbf{u_j} + b^j_{\text{eq}}\right) + \mu^i\left(A_{\text{in}}\mathbf{u_i} - b_{\text{in}}\right) \\
=& \arg\min_{\mathbf{u_i}} \mathbf{u}_i{}^\intercal H_i\mathbf{u}_i + \left(h_i^\intercal + \lambda^{(i,j)}\left(A^i_{\text{eq}} - A^j_{\text{eq}}\right) + \mu^i A_{\text{in}}\right)\mathbf{u_i} \\
\implies& \frac{\partial}{\partial \mathbf{u}_i}\left\{\mathbf{u}_i{}^\intercal H_i\mathbf{u}_i + \left(h_i^\intercal + \lambda^{(i,j)}\left(A^i_{\text{eq}} - A^j_{\text{eq}}\right) + \mu^i A_{\text{in}}\right)\mathbf{u_i}\right\} = 0 \\
\Leftrightarrow& \; 2H_i\mathbf{u}_i + \left(h_i^\intercal + \lambda^{(i,j)}\left(A^i_{\text{eq}} - A^j_{\text{eq}}\right) + \mu^i A_{\text{in}}\right) = 0 \\
\Leftrightarrow& \; = \mathbf{u}_i^* = -\frac{1}{2}H_i^{-1}\left(h_i^\intercal + \lambda^{(i,j)}\left(A^i_{\text{eq}} - A^j_{\text{eq}}\right) + \mu^i A_{\text{in}}\right)
\end{aligned} \tag{19}$$

An iteration of these four sub-problems will return the optimal vectors $\mathbf{u_1}^*$, $\mathbf{u_2}^*$, $\mathbf{u_3}^*$ and $\mathbf{u_4}^*$. For each dual variable, the master problem:

$$\sup_{\mu\geq 0,\lambda} d(\lambda,\mu) \tag{20}$$

can be equivalently updated via a projected subgradient method [4]. The subgradient update is defined in accordance to the discussion had previously for the toy example:

$$\begin{aligned}
\lambda^{(i,j)}_{k+1} &= \lambda^{(i,j)}_{k+1} + \alpha_k\left(A^i_{\text{eq}}\mathbf{u_i}^* + b^i_{\text{eq}} - A^j_{\text{eq}}\mathbf{u_j}^* - b^j_{\text{eq}}\right) \\
\mu^i_{k+1} &= \mathcal{P}_{\mathbb{R}^+_0}\left\{\mu^i_k + \alpha_k\left(A_{\text{in}}\mathbf{u_i}^* - b_{\text{in}}\right)\right\}
\end{aligned} \tag{21}$$

Let $\alpha_k = \alpha$ be a constant step size. The final piece of the algorithm here is the stopping criteria. Since realistically, it cannot be expected that a centralized solution will be available

to compare against, stopping was decided based on fulfillment of the equality constraints. Let $\Delta x_f = \max \|x_i^k(T_{\text{final}}) - x_j^k(T_{\text{final}})\|$, the norm of the largest difference between two final states for consecutive agents at time step k. Then the process is said to have converged when $\Delta x_f \leq \varepsilon$, where ε is a decision variable corresponding to the residual of the process.

The incremental subgradient updates therefore consist of the iterative solution of four optimization problems with fixed dual variables, followed by the update of said dual variables based on the subgradients and the optimal primal values. This scheme can be implemented in a distributed fashion since each node only needs information from the neighbouring nodes in order to update its own dual variables. Further, this update scheme was implemented in a cyclic manner, for simplicity. The whole process can be described as in Algorithm 1.

Algorithm 1 Projected Subgradient Method

Initialize α, ε, $\lambda_0^{(i,j)}$ and μ_0^i
k:=0
while $\Delta x_f > \varepsilon$ **do**
    $\Delta x_f := 0$
    for i := 1 to N **do**
        Compute $\mathbf{u_i}^*$ according to (18)-(19) ▷ Analytically or via `quadprog(·)`
        Compute $\lambda_{k+1}^{(i,j)}$ according to (21).
        Compute μ_{k+1}^i according to (21)
        Compute $x_i^k(T_{\text{final}})$ according to (5)
        j := 1 + mod(i,4)
        Compute $\Delta x_f^{(i,j)} := \|x_i^k(T_{\text{final}}) - x_j^k(T_{\text{final}})\|$
        if $\Delta x_f^{(i,j)} \geq \Delta x_f$ **then**
            $\Delta x_f := \Delta x_f^{(i,j)}$
        end if
        k := k+1
    end for
end while

For comparison purposes, the centralized solution to this problem is easily obtained by considering the stacked version of the problem:

$$\begin{aligned} \min_{\mathbf{u}} \quad & \mathbf{u}^\intercal H \mathbf{u} + h^\intercal \mathbf{u} \\ \text{subject to} \quad & \bar{A}_{\text{eq}} \mathbf{u} = b_{\text{eq}} \\ & \bar{A}_{\text{in}} \mathbf{u} \leq b_{\text{in}} \end{aligned} \tag{22}$$

Where the newly introduced matrices are defined in 23. Note that, in this case, there is no difference at all between the inclusion (or not) of the last equality constraint, which was kept merely for consistency purposes.

The solution to this optimization problem can be easily computed using the `Matlab` Optimization Toolbox. The `quadprog(·)`, for instance, can be used to obtain the optimal solution

to all inputs at once, in this formulation.

$$H = \text{diag}(H_1, H_2, H_3, H_4) \quad , \quad h = \begin{bmatrix} h_1^\intercal & h_2^\intercal & h_3^\intercal & h_4^\intercal \end{bmatrix}^\intercal$$

$$\bar{A}_{\text{eq}} = \begin{bmatrix} A_{\text{eq}}^1 & -A_{\text{eq}}^2 & 0 & 0 \\ 0 & A_{\text{eq}}^2 & -A_{\text{eq}}^3 & 0 \\ 0 & 0 & A_{\text{eq}}^3 & -A_{\text{eq}}^4 \\ -A_{\text{eq}}^1 & 0 & 0 & A_{\text{eq}}^4 \end{bmatrix} \quad , \quad \bar{b}_{\text{eq}} = \begin{bmatrix} b_{\text{eq}}^2 - b_{\text{eq}}^1 \\ b_{\text{eq}}^3 - b_{\text{eq}}^2 \\ b_{\text{eq}}^4 - b_{\text{eq}}^3 \\ b_{\text{eq}}^1 - b_{\text{eq}}^4 \end{bmatrix} \tag{23}$$

$$\bar{A}_{\text{in}} = \begin{bmatrix} A_{\text{in}} & 0 & 0 & 0 \\ 0 & A_{\text{in}} & 0 & 0 \\ 0 & 0 & A_{\text{in}} & 0 \\ 0 & 0 & 0 & A_{\text{in}} \end{bmatrix} \quad , \quad \bar{b}_{\text{in}} = \begin{bmatrix} b_{\text{in}} \\ b_{\text{in}} \\ b_{\text{in}} \\ b_{\text{in}} \end{bmatrix}$$

1.1.1 Results and Analysis

Comparing the results of the projected subgradient method based on dual decomposition, in Figure 1 to the Centralized solution in Figure 2, it's possible to conclude that the incremental method converges nicely to the same results as the centralized problem. Each figure shows four separate sub-plots, each dedicated to a single component of the state $x(t)$. Each sub-plot then shows the evolution of that component for all four aircraft.

Note that both solutions are essentially identical. Further, note that, in both cases, at the final time $T_{\text{final}} = 5$, all four aircraft have the same set of states x_f, as intended. The consensus state is:

$$x_f = \begin{bmatrix} -1.9556 & -1.6274 & -0.8206 & -0.9729 \end{bmatrix}^\intercal \tag{24}$$

Note, however, that the condition number of the resulting matrix H is very large. This is easily obtained by calculating the SVD of H and taking the quotient between the maximum and minimum singular values.

$$\kappa(H) = \frac{\sigma_{\max}(H)}{\sigma_{\min}(H)} \approx 8.27 \times 10^{12} \tag{25}$$

This means that any numerical solution found is highly susceptible to inaccuracies. As a footnote, attempting to solve this problem with `quadprog(·)` returned a different solution than `CVX`. `CVX` itself would return different solutions depending on the solver used. Attempting to use `YALMIP` returned similarly varying results depending on the solver. The choice was ultimately made to use `quadprog(·)` exclusively for the remainder of this report, since the solutions obtained from solving the centralized problem with this solver matched those of solving the decentralized problem analytically, as laid out in equation 19.

Having proved the correctness of the decentralized problem by comparison to the centralized solution, one can now look closed at some aspects of the implementation of the distributed algorithm.

One of the interesting aspect to analyse is the speed of convergence of the algorithm. The influence of step size will be analysed in the next section. Of note here is the use of the fourth

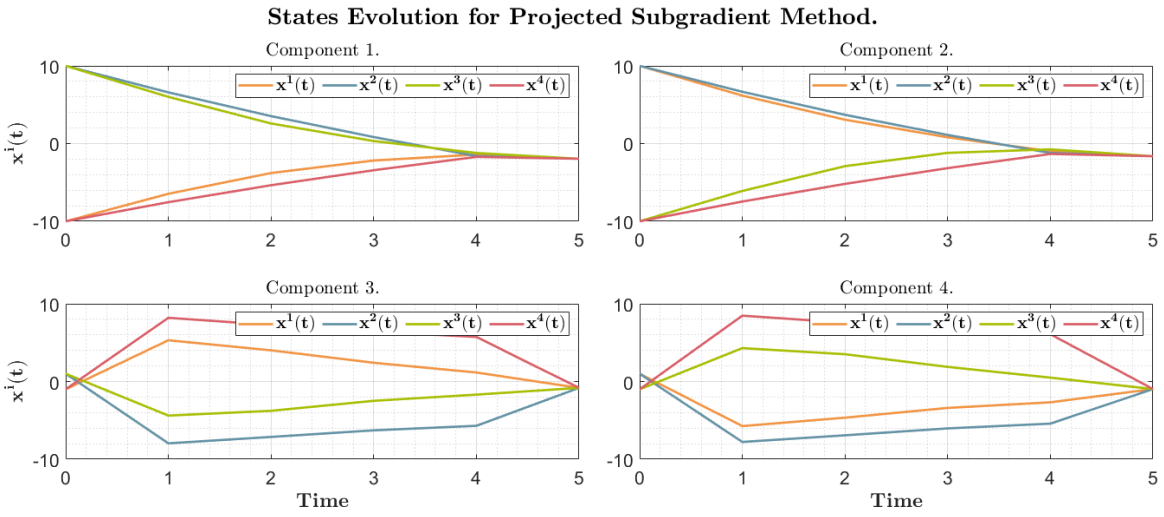

Figure 1: States Evolution for Projected Subgradient Method.

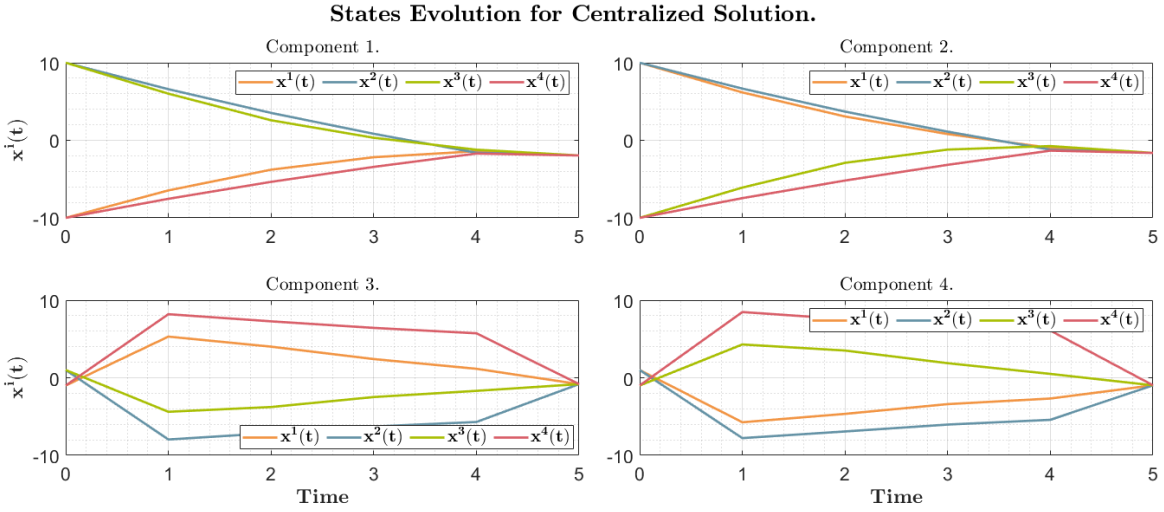

Figure 2: States Evolution for Centralized Solution.

equality constraint $x_f^4 = x_f^1$. As previously alluded to, this is not a necessary constraint for convergence, but it does improve the speed of said convergence, as visible in Figure 3. On this topic, note that this added constraint merely closes the cycle such that each aircraft updates its primal and dual variables based on two other neighbours. It's to be expected that further increasing the amount of neighbours each node communicates with would further increase the speed of convergence.

1.2 Effect of Step Size and Update Sequence

This section takes a look at a practical aspect of the subgradient update critical to the convergence of the algorithm. Two different aspects will be analysed, pertaining to the step size α_k used in the algorithm, and secondly, to the effect of normalizing the gradient when applying these methods.

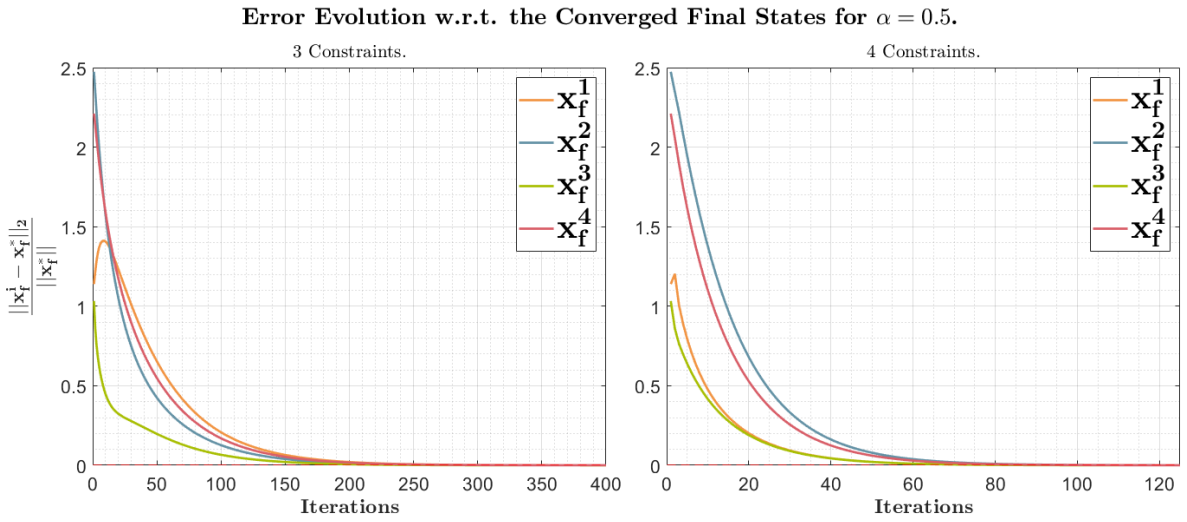

Figure 3: Error between the final states of each aircraft and the centralized solution. Step-size $\alpha = 0.5$ and gradient not-normalized. Three equality constraints (left) vs. four equality constraints (right).

1.2.1 Fixed Step-size and Normalized Subgradient

The only scheme where the update steps are truly fixed happens if both subgradient is normalized and the step-size itself is fixed. In essence, this means the algorithm will always take a step of fixed size α in the direction of the subgradient.

$$\begin{aligned} \lambda_{k+1}^{(i,j)} &= \lambda_{k+1}^{(i,j)} + \alpha \frac{A_{\text{eq}}^i \mathbf{u_i}^* + b_{\text{eq}}^i - A_{\text{eq}}^j \mathbf{u_j}^* - b_{\text{eq}}^j}{\| A_{\text{eq}}^i \mathbf{u_i}^* + b_{\text{eq}}^i - A_{\text{eq}}^j \mathbf{u_j}^* - b_{\text{eq}}^j \|} \\ \mu_{k+1}^i &= \mathcal{P}_{\mathbb{R}_0^+} \left\{ \mu_k^i + \alpha \frac{A_{\text{in}} \mathbf{u_i}^* - b_{\text{in}}}{\| A_{\text{in}} \mathbf{u_i}^* - b_{\text{in}} \|} \right\} \end{aligned} \tag{26}$$

The result is a fast convergence close to the optimum, since the step size is always relatively large, and doesn't get smaller with time. However, since the algorithm is incapable of reaching the true optimum, since the fixed step size won't allow for that, resulting in a large residual, which can only be remedied by taking a smaller step size. Figure 4 shows how the normalized error between the final states of each agent and the centralized solution evolves over iterations for two different values of step-size.

Both solutions converge quickly to a neighbourhood of the optimum. However, the fixed step size reflects itself on the zig-zag pattern visible in the solution, as the algorithm is incapable of getting any closer to the solution. A smaller step size is slower but allows for less residual error and a smaller overall distance to the optimum, as seen in the comparison between $\alpha = 1$ and $\alpha = 2$. Note that $\alpha = 1$ converges slower but comes closer to an error of 10^-1, and the oscillations have less amplitude as well.

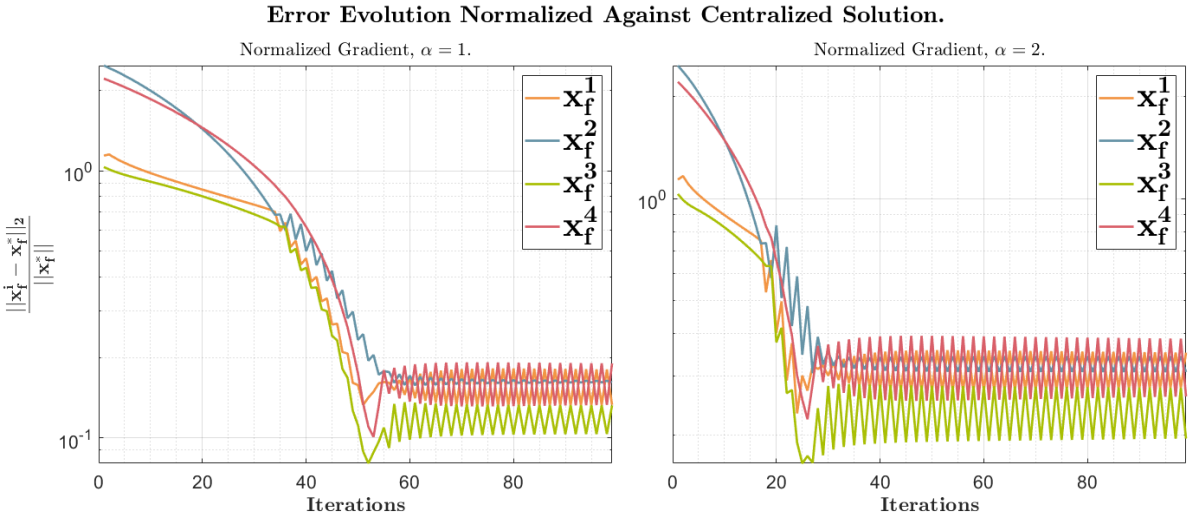

Figure 4: Error between the final states of each aircraft and the centralized solution. Step-sizes $\alpha = 1$ (left) and $\alpha = 2$ (right).

1.2.2 Fixed Step-size and Subgradient Not Normalized

Having the gradient not be normalized effectively means the algorithm will be allowed to vary its step size, despite α being fixed, in a manner proportional to the magnitude of the constraint violation. As the algorithm approaches an optimum, since the problem is strongly convex, the magnitude of the mismatch becomes smaller and the subgradient vanishes to zero. Hence, convergence is achieved through the diminishing magnitude of the subgradient.

$$\begin{aligned} \lambda_{k+1}^{(i,j)} &= \lambda_{k+1}^{(i,j)} + \alpha_k \left(A_{\text{eq}}^i \mathbf{u_i}^* + b_{\text{eq}}^i - A_{\text{eq}}^j \mathbf{u_j}^* - b_{\text{eq}}^j \right) \\ \mu_{k+1}^i &= \mathcal{P}_{\mathbb{R}_0^+} \left\{ \mu_k^i + \alpha_k \left(A_{\text{in}} \mathbf{u_i}^* - b_{\text{in}} \right) \right\} \end{aligned} \tag{27}$$

Figure 5 shows the effect varying the multiplier α associated to the diminishing gradient. As expected, convergence is faster for a larger value of α. For the case of $\alpha = 1$, the solution seems to stabilize around for a normalized error value of just under 10^{-4}. This is likely due to numerical issues caused by the conditioning number of the matrices involved in both the subgradient algorithm, and in obtaining the centralized solution which is being used as a point of comparison.

One downside of this process is the chance of using an α value too large, which causes the algorithm to become unstable. Table 1, gives a summary of the behavior of this method for various fixed α values.

1.2.3 Varying Step-size and Normalized Subgradient

In this scheme, the sub-gradient is once again normalized, meaning the magnitude of the constraint violation has no effect on the convergence of the algorithm. Instead, steps of varying size α_k are always taken in the direction of the gradient, and convergence towards the optimum

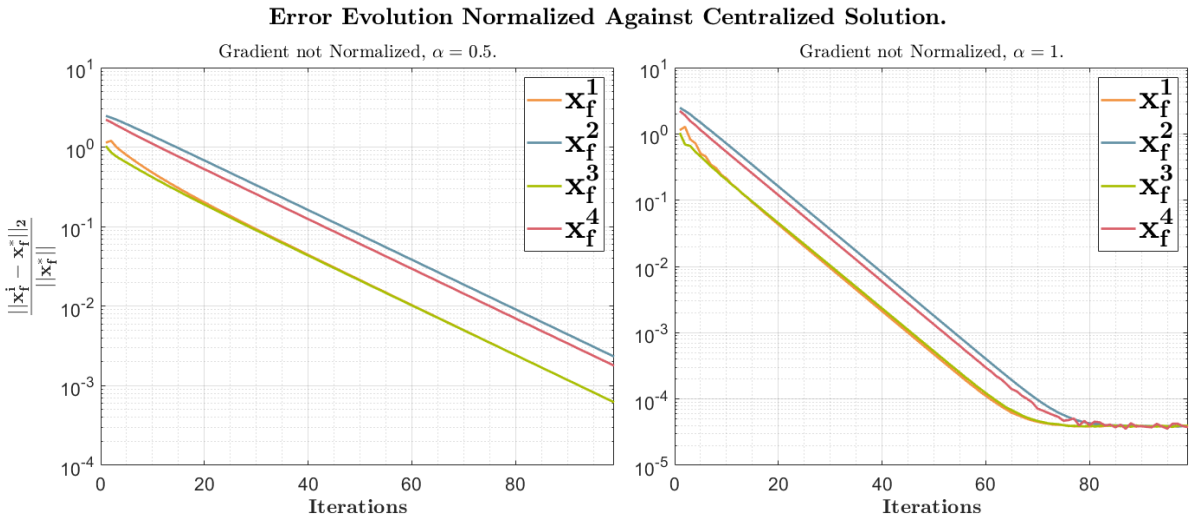

Figure 5: Error between the final states of each aircraft and the centralized solution. Step-sizes $\alpha = 0.5$ (left) and $\alpha = 1$ (right).

α	0.01	0.1	0.5	1	1.1
Iterations	6406	639	126	62	NaN

Table 1: Summary of speed of convergence measured in iterations, depending on step-size and number of equality constraints. For all results, the stopping criteria was $\varepsilon = 10^{-4}$. NaN indicates a diverging solution.

is given by the diminishing step-size α.

$$\begin{aligned} \lambda_{k+1}^{(i,j)} &= \lambda_{k+1}^{(i,j)} + \alpha \frac{A_{\text{eq}}^i \mathbf{u_i}^* + b_{\text{eq}}^i - A_{\text{eq}}^j \mathbf{u_j}^* - b_{\text{eq}}^j}{\| A_{\text{eq}}^i \mathbf{u_i}^* + b_{\text{eq}}^i - A_{\text{eq}}^j \mathbf{u_j}^* - b_{\text{eq}}^j \|} \\ \mu_{k+1}^i &= \mathcal{P}_{\mathbb{R}_0^+} \left\{ \mu_k^i + \alpha \frac{A_{\text{in}} \mathbf{u_i}^* - b_{\text{in}}}{\| A_{\text{in}} \mathbf{u_i}^* - b_{\text{in}} \|} \right\} \end{aligned} \tag{28}$$

The choice of sequence for α_k is crucial in this situation. Convergence is only guaranteed is the sequence is square summable. One possible option here is to choose $\alpha = \frac{c}{k}$, with $c \geq 0$. Figure 6 shows the results for two possible constants, c. As visible, for $c = 1$, convergence occurs, but is exceedingly slow, since the individual steps α_k become small relatively quickly. Using a larger constant $c = 10$ helps with the vanishing step size. Note, however, that larger constant values c are actually prejudicial. Similarly to the previous section, large initial steps with $c > 10$ cause the algorithm to diverge before convergence occurs. Unlike the latter section, however, the algorithm never diverges completely, and convergence happens eventually, since the square summable property still holds.

Take into account how much slower this method is when compared to a fixed α where the step size is dictated by the magnitude of the subgradient. To address this problem, a different step size sequence can be attempted.

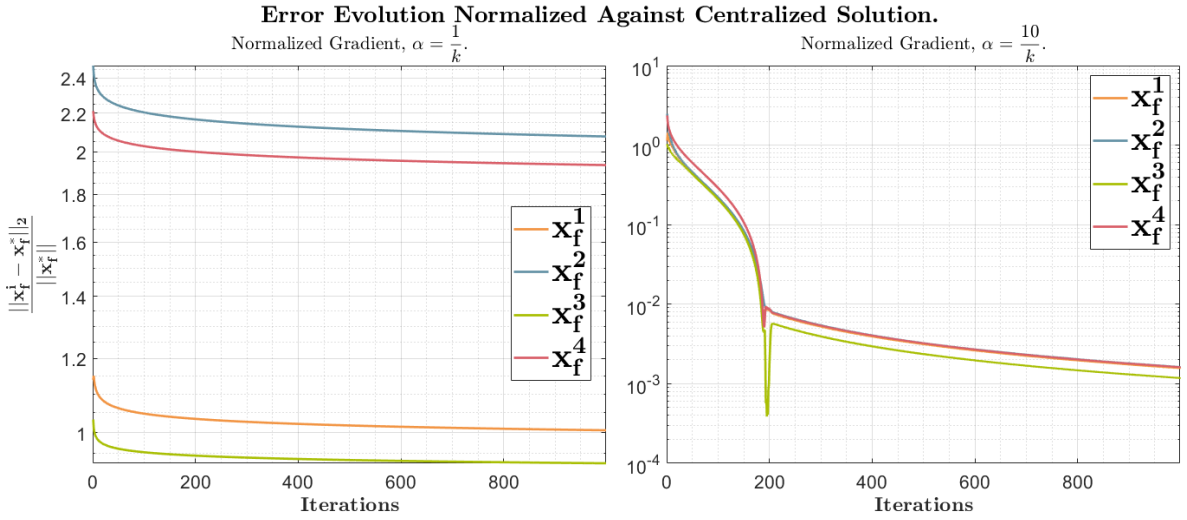

Figure 6: Error between the final states of each aircraft and the centralized solution. Step-sizes $c = 1$ (left) and $c = 10$ (right).

1.2.4 Varying Step-size and Subgradient Not Normalized

Finally, an attempt can be made at allowing both the subgradient magnitude and the α_k step-size to vary. Again, convergence occurs, and seemingly faster than the approach attempted

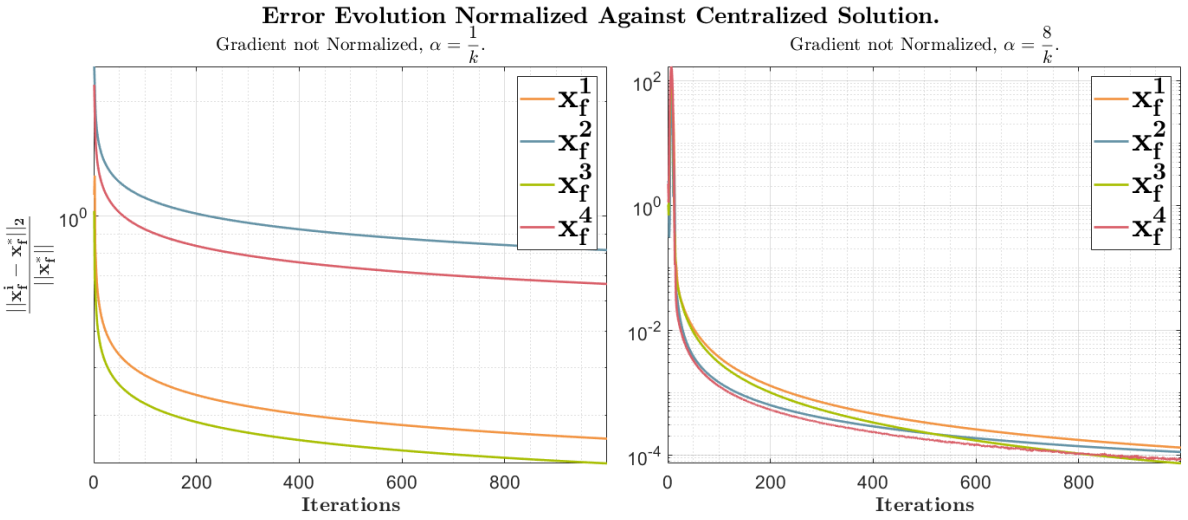

Figure 7: Error between the final states of each aircraft and the centralized solution. Step-sizes $c = 1$ (left) and $c = 8$ (right).

in the previous section, as visible in Figure 7. The reason for this is the initial magnitude of the sub-gradient increasing the convergence at the start.

Speed of convergence here may depend on the gradient magnitude, which, in this case is beneficial, but may not always be the case (if, for instance, the resulting magnitude is always smaller than 1).

1.2.5 Analysis

From the four variants attempted, the one which converged to the solution best was presented in section 1.2.2. Using the magnitude of the gradient seems to a good choice to guarantee fast convergence of the algorithm. The choice is then made to use this scheme going forward.

Sections 1.2.3 and 1.2.4 show other options which are guaranteed to converge as long as the step size sequence α_k is square summable.

The variant shown in 1.2.1 has problems since convergence is fast but only guaranteed to be in a neighbourhood of the optimum.

1.3 Nesterov's Accelerated Subgradient Method

In this section, the objective is to implement an accelerated version (Nesterov method) of the subgradient updates. To do so, the basis is a paper by Giselsson et al. [1].

One can start by reorganizing the Lagrangian in 16. Defining:

$$\lambda = \begin{bmatrix} \lambda^{(1,2)} \\ \lambda^{(2,3)} \\ \lambda^{(3,4)} \\ \lambda^{(4,1)} \end{bmatrix} \in \mathbb{R}^{\ell \times 1}, \quad \mu = \begin{bmatrix} \mu^1 \\ \mu^2 \\ \mu^3 \\ \mu^4 \end{bmatrix} \in \mathbb{R}^{m \times 1}, \quad \mathbf{u} = \begin{bmatrix} \mathbf{u_1} \\ \mathbf{u_2} \\ \mathbf{u_3} \\ \mathbf{u_4} \end{bmatrix} \in \mathbb{R}^{N \cdot n_u \times 1} \tag{29}$$

The Lagrangian can be re-written more compactly:

$$L(\mathbf{u}, \lambda, \mu) = \mathbf{u}^\intercal H \mathbf{u} + h^\intercal \mathbf{u} + \lambda^\intercal \bar{A}_{\text{eq}} \mathbf{u} - \lambda^\intercal \bar{b}_{\text{eq}} + \mu^\intercal \bar{A}_{\text{in}} \mathbf{u} - \mu^\intercal \bar{b}_{\text{in}} \tag{30}$$

Bundling all dual variables in a vector $z = [\lambda^\intercal \ \ \mu^\intercal]^\intercal$, the following expression for the Lagrangian can be derived:

$$\begin{aligned} L(\mathbf{u}, \lambda, \mu) &= \mathbf{u}^\intercal H \mathbf{u} + \left(h^\intercal + \lambda^\intercal \bar{A}_{\text{eq}} + \mu^\intercal \bar{A}_{\text{in}}\right) \mathbf{u} - \lambda^\intercal \bar{b}_{\text{eq}} - \mu^\intercal \bar{b}_{\text{in}} \\ &= \mathbf{u}^\intercal H \mathbf{u} + \left(h + \bar{A}_{\text{eq}}^\intercal \lambda + \bar{A}_{\text{in}}^\intercal \mu\right)^\intercal \mathbf{u} - \bar{b}_{\text{eq}}^\intercal \lambda - \bar{b}_{\text{in}}^\intercal \mu \\ &= \mathbf{u}^\intercal H \mathbf{u} + \left(h + \underbrace{\begin{pmatrix} \bar{A}_{\text{eq}}^\intercal & \bar{A}_{\text{in}}^\intercal \end{pmatrix}}_{\bar{A}} \begin{pmatrix} \lambda \\ \mu \end{pmatrix} \right)^\intercal \mathbf{u} - \underbrace{\begin{pmatrix} \bar{b}_{\text{eq}}^\intercal & \bar{b}_{\text{in}}^\intercal \end{pmatrix}}_{\bar{b}^\intercal} \begin{pmatrix} \lambda \\ \mu \end{pmatrix} \\ &= \mathbf{u}^\intercal H \mathbf{u} + \left(h + \bar{A} z\right)^\intercal \mathbf{u} - \bar{b}^\intercal z \end{aligned} \tag{31}$$

The dual master problem in 17 is recast as:

$$\sup_z d(z) = \sup_z \left\{ \inf_{\mathbf{u}} \left[\mathbf{u}^\intercal H \mathbf{u} + \left(h + \bar{A} z\right)^\intercal \mathbf{u} \right] - \bar{b}^\intercal z \right\} \tag{32}$$

Note that, since the problem is fully unconstrained, the solution to $\inf_{\mathbf{u}}[\cdot]$ can be obtained

analytically:

$$\begin{aligned}
\inf_{\mathbf{u}} \left[\mathbf{u}^\intercal H \mathbf{u} + \left(h + \bar{A}z\right)^\intercal \mathbf{u}\right] \implies & \frac{\partial}{\partial \mathbf{u}} \left[\mathbf{u}^\intercal H \mathbf{u} + \left(h + \bar{A}z\right)^\intercal \mathbf{u}\right] = 0 \\
& \Leftrightarrow \mathbf{u}^* = -\frac{1}{2} H^{-1} \left(A^\intercal z + h\right) \\
\inf_{\mathbf{u}} \left[\mathbf{u}^\intercal H \mathbf{u} + \left(h + \bar{A}z\right)^\intercal \mathbf{u}\right] &= \mathbf{u}^{*\intercal} H \mathbf{u}^* + \left(h + \bar{A}z\right)^\intercal \mathbf{u}^* \\
&= -\frac{1}{4} \left(\bar{A}^\intercal z + h\right)^\intercal H^{-1} \left(\bar{A}^\intercal z + h\right)
\end{aligned} \tag{33}$$

Replacing the newfound expression in the dual problem

$$\begin{aligned}
\sup_z d(z) &= \sup_z \left\{ -\frac{1}{4} \left(\bar{A}^\intercal z + h\right)^\intercal H^{-1} \left(\bar{A}^\intercal z + h\right) - \bar{b}^\intercal z \right\} \\
&= \inf_z \left\{ \frac{1}{4} \left(\bar{A}^\intercal z + h\right)^\intercal H^{-1} \left(\bar{A}^\intercal z + h\right) + \bar{b}^\intercal z \right\} = \inf_z f(z)
\end{aligned} \tag{34}$$

The gradient needed to implement the accelerated subgradient method can be obtained analytically:

$$\nabla f(z) = \frac{1}{2} \bar{A} H^{-1} (\bar{A}^\intercal z + h) + \bar{b} \tag{35}$$

To find the Lipschitz constant bounding the gradient, consider two feasible dual vectors x and y.

$$\begin{aligned}
\|\nabla f(x) - \nabla f(y)\|_2 &= \left\| \left(\frac{1}{2} \bar{A} H^{-1} (\bar{A}^\intercal x + h) + \bar{b} \right) - \left(\frac{1}{2} \bar{A} H^{-1} (\bar{A}^\intercal y + h) + \bar{b} \right) \right\|_2 \\
&= \frac{1}{2} \|\bar{A} H^{-1} \bar{A}^\intercal (x - y)\|_2 \leq \frac{1}{2} \|\bar{A} H^{-1} \bar{A}^\intercal\|_2 \|x - y\|_2 \\
&= L \|x - y\|_2 \implies L = \frac{1}{2} \|\bar{A} H^{-1} \bar{A}^\intercal\|_2
\end{aligned} \tag{36}$$

Implementing Nesterov's method from here is a rather simple task. Defining the iterations as:

$$\begin{aligned}
v^k &= z^k + \alpha_k (z^k - z^{k-1}) \\
z^{k+1} &= \mathcal{P}_Z \left(v^k - \frac{1}{L} \nabla f(v^k) \right)
\end{aligned} \tag{37}$$

Parameter α_k is an update sequence which varies in the literature and which will be explored further in the next section. The corresponding procedure is explained in Algorithm 2. Note that, as implemented, the updates are centralized and an iteration is executed after all four agents have executed their respective tasks. However, taking into account the block diagonal structure of H, it's possible to partition the update expression and perform the updates on the dual variables separately in a distributed manner.

1.3.1 Results and Analysis

The results shown in Figure 8 consider two different values for the sequence α_k. Sequence α_k^1 is as indicated in the slides [4] while sequence α_k^2 is used in the reference paper [1]. For an iteration k, each is defined as:

$$\alpha_k^1 = \frac{1 - \sqrt{k}}{1 + \sqrt{k}} \quad , \quad \alpha_k^2 = \frac{k - 1}{k + 2} \tag{38}$$

Algorithm 2 Nesterov's Accelerated Subgradient Method

Initialize L, ε, $\lambda_0^{(i,j)}$, μ_0^i and $z^0 = z^{-1}$
k:=0
while $\Delta x_f > \varepsilon$ **do**
    $\Delta x_f = 0$
    for i := 1 to N **do**
        Compute $\mathbf{u_i}^*$ according to (18)-(19) ▷ Compute Optimal Primal Variables.
        Compute $x_i^k(T_{\text{final}})$ according to (5)
        j = 1 + mod(i,4)
        Compute $\Delta x_f^{(i,j)} := \|x_i^k(T_{\text{final}}) - x_j^k(T_{\text{final}})\|$ ▷ Check for Convergence.
        if $\Delta x_f^{(i,j)} \geq \Delta x_f$ **then**
            $\Delta x_f := \Delta x_f^{(i,j)}$
        end if
    end for
    Compute v^k according to (37) ▷ Implement Dual Variable Update.
    Compute z^{k+1} according to (37)
    Assign $\lambda_{k+1} := z^{k+1}(1 : \ell)$
    Assign $\mu^{k+1} := z^{k+1}(\ell + 1 : \text{end})$
    k := k+1
end while

In this case, convergence is much faster if sequence α_k^2 is used. However, it's still slower than the best results obtained in Section 1.2.2. To attempt to remedy this, an additional factor β was multiplied by the update sequence α_k^2. Figure 9 shows the results of this attempt. The fastest convergence was achieved for $\beta = 0.75$, improving on the speed of convergence of all methods shown previously. Convergence to an error $\varepsilon < 10^-4$ occurs in less than 40 iterations. Note the rather odd behaviour of the error sequence when using the update sequence α_k^2. Though strange, it does not appear to be incorrect. Since Nesterov's method is not a descent method, the error is guaranteed to go to zero asymptotically, but it's not guaranteed to be strictly decreasing, as shown.

1.4 Combined Consensus/Subgradient Scheme

Implementation of the consensus strategy requires a change in the approach to the problem at hand. Up until now, the final state x_f has been replaced by a function of the inputs of each of the four aircraft. Implementation of the consensus strategy on the final state, however, requires that each aircraft has a (possibly different) fixed final state, and that, through a process of consensus, the four agents converge towards the optimal state.

Implementation of this problem follows a simple procedure. One can start by separating

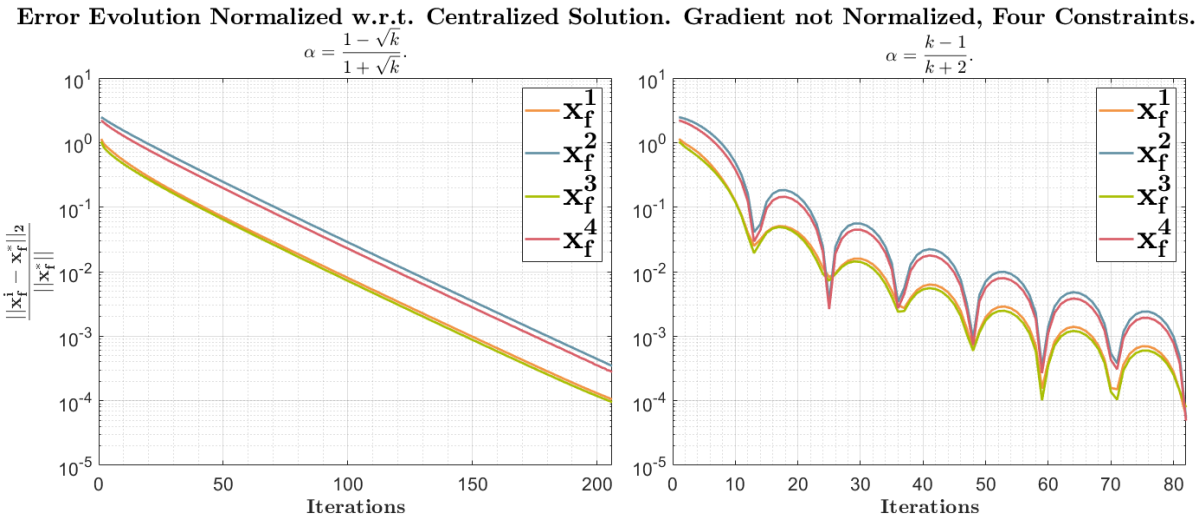

Figure 8: Error between the final states of each aircraft and the centralized solution.

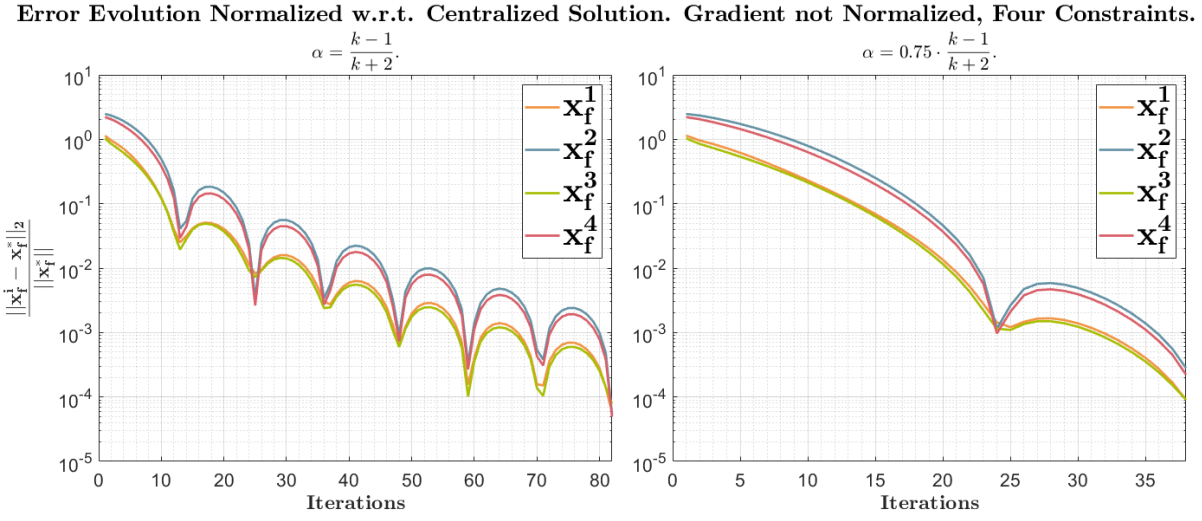

Figure 9: Error between the final states of each aircraft and the centralized solution.

the problem into four sub-problems. For $i = 1, 2, 3, 4$, each sub-problem is defined as:

$$\begin{aligned} \min_{\mathbf{u}_i} \quad & \mathbf{u}_i^\intercal H_i \mathbf{u}_i + h_i^\intercal \mathbf{u}_i \\ \text{subject to} \quad & A_{\text{eq}}^i \mathbf{u}_i + b_{\text{eq}}^i = x_f^i \\ & A_{\text{in}}^i \mathbf{u}_i \leq b_{\text{in}}^i \end{aligned} \tag{39}$$

To proceed with the consensus method, first it's necessary to find a subgradient of each consensus x_f^i variable with respect to its sub-problem. As shown previously in Assignment 3, a subgradient can be found by taking the negative of the optimal dual variables associated to the problem. In this case, if we write the Lagrangian of each subproblem as:

$$L^i(\mathbf{u}_i, \lambda^i, \mu^i) = \mathbf{u}_i^\intercal H_i \mathbf{u}_i + h_i^\intercal \mathbf{u}_i + \lambda^{i\intercal} \left(A_{\text{eq}}^i \mathbf{u}_i + b_{\text{eq}}^i - x_f^i \right) + (\mu^i)^\intercal \left(A_{\text{in}}^i \mathbf{u}_i - b_{\text{in}}^i \right) \tag{40}$$

If $p(x_f^i)$ is the optimal value of the convex optimization problem i, and $(\lambda^i)^*$ are the optimal dual

variables associated with the constraint $\left(A^i_{\text{eq}}\mathbf{u}_i - b^i_{\text{eq}} = x^i_f\right)$, then $-(\lambda^i)^*$ will be a subgradient of $p(\cdot)$ at x^i_f. A brief proof of this is further outlined in [3].

Finding the optimal value of the dual variables for each problem can be done in multiple ways. One way is to solve to relax the equality constraint of the problems in 41, initialize the corresponding dual variables $\lambda^i = \lambda^i_0$ and solve the problem of:

$$\begin{aligned} \min_{\mathbf{u}_i} \quad & \mathbf{u}_i^\intercal H_i \mathbf{u}_i + h_i^\intercal \mathbf{u}_i + \lambda^{i\intercal}\left(A^i_{\text{eq}}\mathbf{u}_i + b^i_{\text{eq}} - x^i_f\right) \\ \text{subject to} \quad & A^i_{\text{in}}\mathbf{u}_i \leq b^i_{\text{in}} \end{aligned} \tag{41}$$

Using the optimal primal variables $\mathbf{u}^*_i$, the dual variables λ^i can the be updated iteratively via subgradient updates (With subgradient $g = \left(A^i_{\text{eq}}\mathbf{u}^*_i + b^i_{\text{eq}} - x^i_f\right)$), similarly to the procedure in Section 1.1.

The resulting process is called Dual Ascent. In testing, while the dual variables always converged to an optimum $(\lambda^i)^*$, it was also a rather slow procedure, taking hundreds of iterations to converge. As such, the choice was made to use the functionality of `quadprog(·)` and extract the dual variables directly from the solver. To get the dual variables corresponding to the equality constraints, the problem to be solved had to be fully constrained, as in 41. Note that the optimal dual variables associated to the equality constraints were the same whether Dual Ascent was used or they we're simply extracted from the solver.

All the ingredients needed in order to apply the Combined Consensus/Subgradient Scheme have now been described, and it's possible to move to implement the algorithm. The generic update procedure is:

$$\theta^i_{k+1} = \mathcal{P}_\Theta\left[\sum_{j=1}^{N}[W^\varphi]_{ij}\left(\theta^j_k - \alpha_k g^j\left(\theta^j_k\right)\right)\right], \quad i = 1, \ldots, N \tag{42}$$

Application of to the problem at hand is straightforward:

$$x^i_f(k+1) = \mathcal{P}_X\left[\sum_{j=1}^{N}[W^\varphi]_{ij}\left(x^j_f(k) + \alpha(k)(\lambda^i)^*\right)\right], \quad i = 1, \ldots, 4 \tag{43}$$

Let $x^{i,\text{loc}}_f(k+1) = x^i_f(k) + \alpha(k)(\lambda^i)^*$, $x^{i,\text{loc}}_f(k) \in \mathbb{R}^{4\times 1}$ represent the local updates of one agent to its own final state. What the consensus method accomplishes is the weighing of the local update $x^{i,\text{loc}}_f(k)$ with the local updates of the aircraft's neighbours, $x^{j,\text{loc}}_f(k)$, via a consensus matrix. The resulting final state for aircraft i at time step $k+1$ is a linear combination of the local updates of itself and the neighbours $x^i_f(k)) = f\left(x^{i,\text{loc}}_f(k), x^{j,\text{loc}}_f(k)\right)$.

The procedure in 43 can be compactly written as:

$$\begin{bmatrix} x^1_f(k) & x^2_f(k) & x^3_f(k) & x^4_f(k)\end{bmatrix}^\intercal = \mathcal{P}_X\left\{W^\varphi\begin{bmatrix} x^{1,\text{loc}}_f(k) & x^{2,\text{loc}}_f(k) & x^{3,\text{loc}}_f(k) & x^{4,\text{loc}}_f(k)\end{bmatrix}^\intercal\right\} \tag{44}$$

The stopping criteria for the algorithm is based on the convergence of all four final states x^i_f. Let $\Delta x^i_f(k) = \|x^i_f(k) - x^i_f(k-1)\|$. Then the stopping criteria was implemented as $\Delta x_f(k) =$

Algorithm 3 Combined Consensus/Subgradient Scheme

Initialize L, ε, $x_f^i(0)$, W, φ, α
k:=0
while $\Delta x_f > \varepsilon$ **do**
    $\Delta x_f = 0$
    for i := 1 to N **do**
        Compute $\mathbf{u_i}^*$ according to (41) ▷ Via `quadprog(·)`
        Compute $x_i^k(T_{\text{final}})$ according to (5)
        j = 1 + mod(i,4)
        Compute $\Delta x_f^{(i,j)} := \|x_i^k(T_{\text{final}}) - x_j^k(T_{\text{final}})\|$ ▷ Check for Convergence
        if $\Delta x_f^{(i,j)} \geq \Delta x_f$ **then**
            $\Delta x_f := \Delta x_f^{(i,j)}$
        end if
    end for
    Compute $x_f^i(k+1)$ according to (43) ▷ Combined Consensus/Subgradient Scheme
    k := k+1
end while

$\max(\Delta x_f^i(k)) \leq \varepsilon$, where the choice was made to set $\varepsilon = 10^{-3}$. Implementing the scheme is now a straightforward task. The description of the implementation can be found in Algorithm 3. It's useful to take a moment to go over the expected results. Depending on the step size, α and on the number of consensus iterations, ϕ, the results can be very different. Theorem 1 in [2] shows that the sequence generated for each x_f^i will have:

$$\liminf_{k\to\infty} f\left(x_k^i\right) \leq f^\star + \frac{\alpha N C^2}{2} + 3NC\beta, \forall i = 1, \ldots, N, \quad \text{if } f^\star < -\infty \tag{45}$$

where $f^\star = f(x_f)$, with x_f optimal value of x_f^* as defined in 24.

The implications of this is theorem are that the consensus solutions will converge to within a ball of radius β of the optimum. In particular, this means that, if α is fixed, the solutions don't need to converge to the same exact value between themselves, and they don't need to converge to the optimum either, $(x_f^1)^* \neq (x_f^2)^* \neq (x_f^3)^* \neq (x_f^4)^* \neq x_f^*$.

However, the theorem also states that the individual solutions will approach themselves and the optimum for large enough values of φ (resulting in a smaller radius β) and small enough step-size α. In the limit $\phi \to \infty$, the process will approach the standard subgradient method, each of the four agent's solutions will be averaged at each time step resulting in $\beta = 0$. If additionally the step-size is square summable (implying in particular that $\alpha_k \to 0$ as $k \to \infty$), then the two results will converge and the process will reach an optimum.

1.4.1 Results and Analysis

This section is concerned with the analysis of the previously presented algorithm for varying parameters α and φ.

All figures below show both solid and dashed (thinner) lines. Solid lines represent the average error of the four agents measured against the centralized solution. The dashed lines of the same color represent each agent. The choice was made to focus on the average of the four errors and not on each specific error, since it simplified the analysis and allowed for the same conclusions.

Considering a fixed step size α and varying the number of consensus iterations φ, Figure 10 is obtained. As expected, as φ increases and the radius of the ball β decreases, the convergence

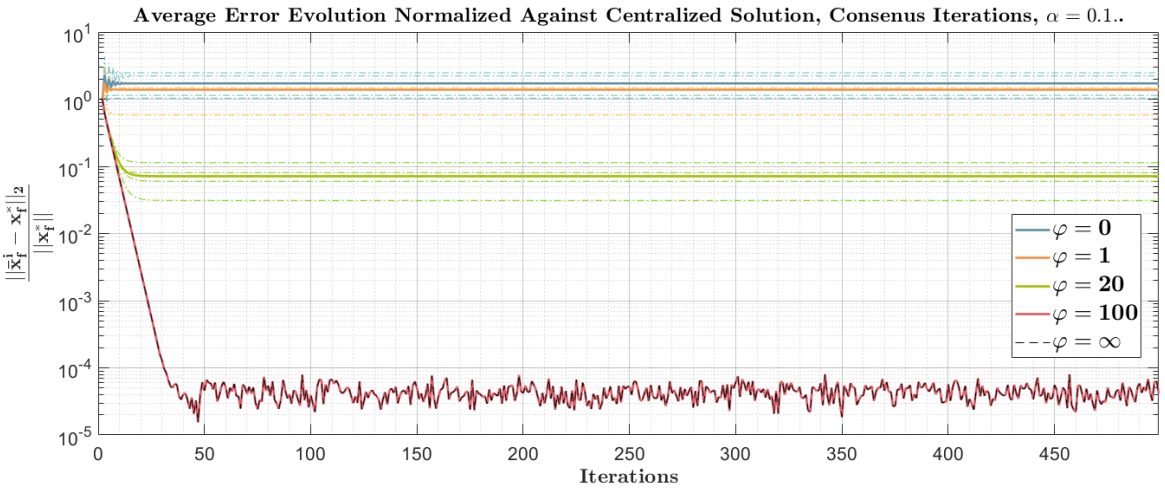

Figure 10: Error between the final states of each aircraft and the centralized solution.

comes closer to the optimum value. For $\varphi = 100$, the solution is virtually indistinguishable from the average subgradient method, represented by $\varphi = \infty$. This is to be expected, since as $\varphi \to \infty$, the consensus method should approach that of the standard subgradient method.

Figure 11 shows the difference different step sizes α can have in the convergence of the algorithm, for a fixed value $\varphi = 20$. Once again, results seem to agree with equation 45. As *alpha* decreases, so does the distance to the optimal result. In both Figures above, the results seem to stabilize after some time. This, once again, makes sense, since both β and α are fixed (β being dictated by φ). As such, the upper limit of

$$\liminf_{k\to\infty} f\left(x_k^i\right) \leq f^\star + \frac{\alpha N C^2}{2} + 3NC\beta, \forall i = 1, \ldots, N, \quad \text{if } f^\star < -\infty \tag{46}$$

remains constant, and the solutions stabilize once they enter that region. To shrink the upper limit, one could either increase φ, performing more consensus iterations as $k \to \infty$, or decrease $\alpha \to 0$ as $k \to \infty$. Figure 12 shows an attempt at the latter approach, where the algorithm is implemented with a vanishing step-size, meaning the upper bound on 45 becomes smaller

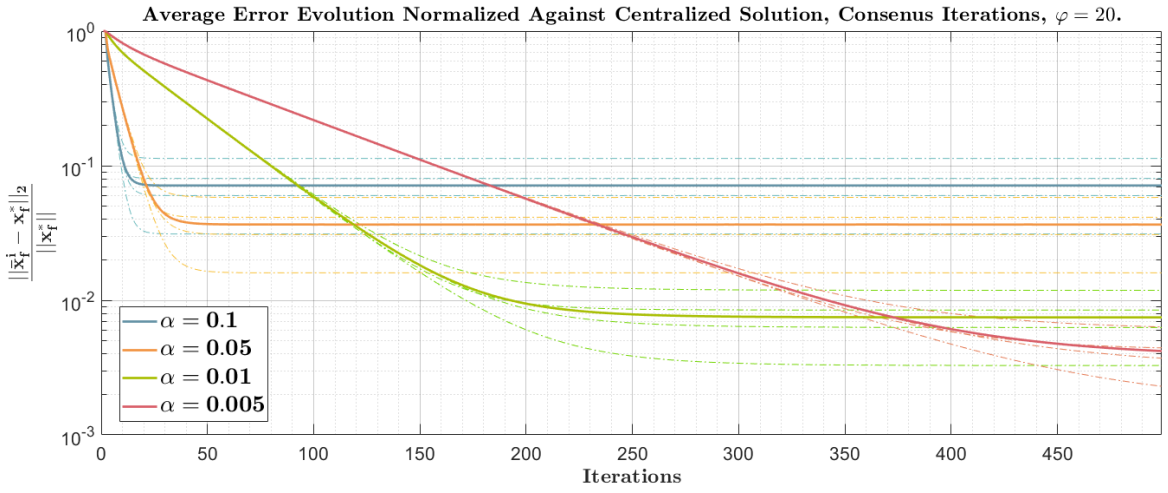

Figure 11: Error between the final states of each aircraft and the centralized solution.

over time. Compared to Figure 10, we see slower convergence because of the step-size becoming smaller as $k \to \infty$. However, the plots keeps a downward trend as the upper bound on $f(x_k^i)$ gets progressively tighter.

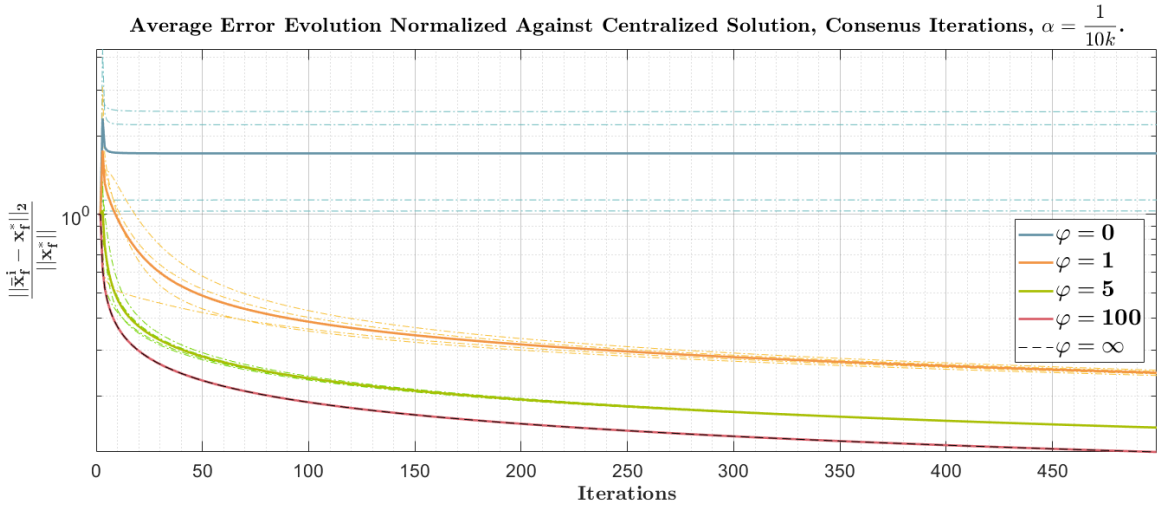

Figure 12: Error between the final states of each aircraft and the centralized solution.

2 Problem 2

2.1 ADMM for Consensus Optimization

Implementation of the Alternating Direction Method of Multipliers relies primarily on the slides available in [4]. ADMM for Consensus Optimization is based on the cyclic update of the primal, consensus and dual variables and is based on the augmented Lagrangian to add robustness to the standard dual approach.

To approach this problem, the first step is to present the augmented Lagrangian which will be used. In this section, only the equality constraints are relaxed, and resulting augmented Lagrangian is:

$$L(\mathbf{u}_i, x_f, \lambda^i) = \sum_{i=1}^{N} \left\{ V(\mathbf{u}_i) + \lambda^{i\intercal} \left(A^i_{\text{eq}} \mathbf{u}_i + b^i_{\text{eq}} - x_f \right) + \frac{\rho}{2} \| A^i_{\text{eq}} \mathbf{u}_i + b^i_{\text{eq}} - x_f \|_2^2 \right\} \tag{47}$$

where $V(\mathbf{u}_i)$ is the optimal solution to the constrained optimization problem:

$$\begin{aligned} V(\mathbf{u}_i) = \; & \min_{\mathbf{u}_i} \mathbf{u}_i^\intercal H_i \mathbf{u}_i + h_i^\intercal \mathbf{u}_i \\ & \text{subject to } A_{\text{in}} \mathbf{u}_i \leq b_{\text{in}} \end{aligned} \tag{48}$$

Note that, unlike in the standard consensus approach, in which each aircraft kept a local copy of x_f and a consensus was agreed between all of them, in this approach, the final state x_f is unique and shared between all agents. The augmented Lagrangian is still clearly separable into four separate optimization problems from which the optimal primal variables can be retrieved. The updates to each of the variables is done according to:

$$\begin{aligned} \mathbf{u}_i^{k+1} &= \arg\min_{\mathbf{u}_i} L^i(\mathbf{u}_i, x_f, \lambda^i) \\ x_f^{k+1} &= \frac{1}{N} \sum_{i=1}^{N} \left\{ \arg\min_{x_f} L^i(\mathbf{u}_i^{k+1}, x_f, \lambda^i) \right\} \\ (\lambda^i)^{k+1} &= (\lambda^i)^k + \rho \left(A^i_{\text{eq}} \mathbf{u}_i^{k+1} + b^i_{\text{eq}} - x_f^{k+1} \right) \end{aligned} \tag{49}$$

The update on $\mathbf{u}_i$ is obtained by solving each individual problem. These new optimal values are then used in an averaging step to update the consensus variable x_f. Finally, both updated variables are used in a standard subgradient method to update the dual variables. Note that the update on x_f can actually be obtained analytically. Dropping all terms independent of x_f:

$$\begin{aligned} \arg\min_{x_f} L^i(\mathbf{u}_i, x_f, \lambda^i) &= \arg\min_{x_f} \left\{ \lambda^{i\intercal} \left(A^i_{\text{eq}} \mathbf{u}_i + b^i_{\text{eq}} - x_f \right) + \frac{\rho}{2} \| A^i_{\text{eq}} \mathbf{u}_i + b^i_{\text{eq}} - x_f \|_2^2 \right\} \\ &= \arg\min_{x_f} \left\{ -\lambda^{i\intercal} x_f + \frac{\rho}{2} \left(A^i_{\text{eq}} \mathbf{u}_i + b^i_{\text{eq}} - x_f \right)^\intercal \left(A^i_{\text{eq}} \mathbf{u}_i + b^i_{\text{eq}} - x_f \right) \right\} \\ &= \arg\min_{x_f} \left\{ -\lambda^{i\intercal} x_f + \rho \left(\frac{1}{2} x_f^\intercal x_f - \left(A^i_{\text{eq}} \mathbf{u}_i + b^i_{\text{eq}} \right) x_f \right) \right\} \end{aligned} \tag{50}$$

Noting that the problem is an unconstrained minimization problem, the optimal x_f is given by:

$$\begin{aligned} & \frac{\partial}{\partial x_f} \left(-\lambda^{i\intercal} x_f + \rho \left(\frac{1}{2} x_f^\intercal x_f + \left(A^i_{\text{eq}} \mathbf{u}_i + b^i_{\text{eq}} \right) x_f \right) \right) = 0 \\ \Leftrightarrow \; & -\lambda^{i\intercal} - \rho \left(A^i_{\text{eq}} \mathbf{u}_i + b^i_{\text{eq}} \right) + \rho \, x_f = 0 \\ \Leftrightarrow \; & x_f = \frac{\lambda^{i\intercal}}{\rho} + \left(A^i_{\text{eq}} \mathbf{u}_i + b^i_{\text{eq}} \right) \end{aligned} \tag{51}$$

Similarly, the update of variables $\mathbf{u}_i$ can be clarified:

$$\begin{aligned} \arg\min_{\mathbf{u}_i} L^i(\mathbf{u}_i, x_f, \lambda^i) &= \arg\min_{x_f} \left\{ \lambda^{i\intercal} \left(A^i_{\text{eq}} \mathbf{u}_i + b^i_{\text{eq}} - x_f \right) + \frac{\rho}{2} \| A^i_{\text{eq}} \mathbf{u}_i + b^i_{\text{eq}} - x_f \|_2^2 \right\} \\ &= \arg\min_{\mathbf{u}_i} \left\{ \lambda^{i\intercal} A_{\text{eq}} \mathbf{u}_i + \frac{\rho}{2} \left(A^i_{\text{eq}} \mathbf{u}_i + b^i_{\text{eq}} - x_f \right)^\intercal \left(A^i_{\text{eq}} \mathbf{u}_i + b^i_{\text{eq}} - x_f \right) \right\} \\ &= \arg\min_{\mathbf{u}_i} \left\{ \lambda^{i\intercal} A^i_{\text{eq}} \mathbf{u}_i + \frac{\rho}{2} \left(\mathbf{u}_i^\intercal {A^i_{\text{eq}}}^\intercal A^i_{\text{eq}} \mathbf{u}_i + 2 \left(b^i_{\text{eq}} - x_f \right)^\intercal A^i_{\text{eq}} \mathbf{u}_i \right) \right\} \end{aligned} \tag{52}$$

Taking in to account the definition of $V(\mathbf{u}_i)$ in 48, the ADMM updates in 49 can therefore be re-written as:

$$\begin{aligned}\mathbf{u}_i^{k+1} = \quad & \arg\min_{\mathbf{u}_i} \ \mathbf{u}_i^{\intercal}\left(H_i + \frac{\rho}{2}{A_{\text{eq}}^i}^{\intercal}A_{\text{eq}}^i\right)\mathbf{u}_i + \left(h_i^{\intercal} + \lambda^{i\intercal}A_{\text{eq}}^i + \rho\left(b_{\text{eq}}^i - x_f\right)^{\intercal}A_{\text{eq}}^i\right)\mathbf{u}_i \\ & \text{subject to } \ A_{\text{in}}\mathbf{u}_i \leq b_{\text{in}}\end{aligned} \tag{53}$$

$$x_f^{k+1} = \frac{1}{N}\sum_{i=1}^{N}\left\{\frac{\lambda^{i\intercal}}{\rho} + \left(A_{\text{eq}}^i\mathbf{u}_i^{k+1} + b_{\text{eq}}^i\right)\right\} \tag{54}$$

$$(\lambda^i)^{k+1} = (\lambda^i)^k + \rho\left(A_{\text{eq}}^i\mathbf{u}_i^{k+1} + b_{\text{eq}}^i - x_f^{k+1}\right) \tag{55}$$

Implementation from this point onward is straightforward. The stopping criteria was defined based solely on the stationary of x_f. Let $\Delta x_f^k = x_f^k - x_f^{k-1}$. The algorithm is assumed to have converged if $\Delta x_f^k \leq \varepsilon$. A summary of the algorithm can be found in 4. The results are shown

Algorithm 4 Alternating Direction Method of Multipliers for Consensus Optimization

1: Initialize ε, ρ, x_f^0, $\left(\lambda^i\right)^0$
2: $k := 0$
3: $\Delta x_f^0 := \infty$
4: **while** $\Delta x_f^k > \varepsilon$ **do**
5: **for** i := 1 to N **do**
6: Compute ${\mathbf{u_i}}^{k+1}$ according to (53)
7: **end for**
8: Compute ${x_f}^{k+1}$ according to (54)
9: **for** i := 1 to N **do**
10: Compute $(\lambda^i)^{k+1}$ according to (55)
11: **end for**
12: Compute $\Delta x_f^{k+1} = x_f^{k+1} - x_f^k$
13: k := k+1
14: **end while**

in figure 13 for a value of $\rho = 1$. Convergence occurs nicely as the ADMM method manages to reduce the error between the consensus variable x_f and the centralized solution to less than 10^{-4} within about 140 iterations. The stabilization around 10^{-4} and subsequent noisy behaviour are likely attributed to the same numerical issues addressed previously.

2.2 Effect of the Penalty Parameter on the Convergence Rate

This section is concerned with the analysis of the results for different values of the penalty parameter ρ. Figure 14 shows the convergence results of the consensus variable for various values of ρ.

Looking at the results, it's obvious that an increase in ρ values is initially advantageous. Convergence rate improves in the range $\rho \in (0, 7]$. However, it's also possible to see that these

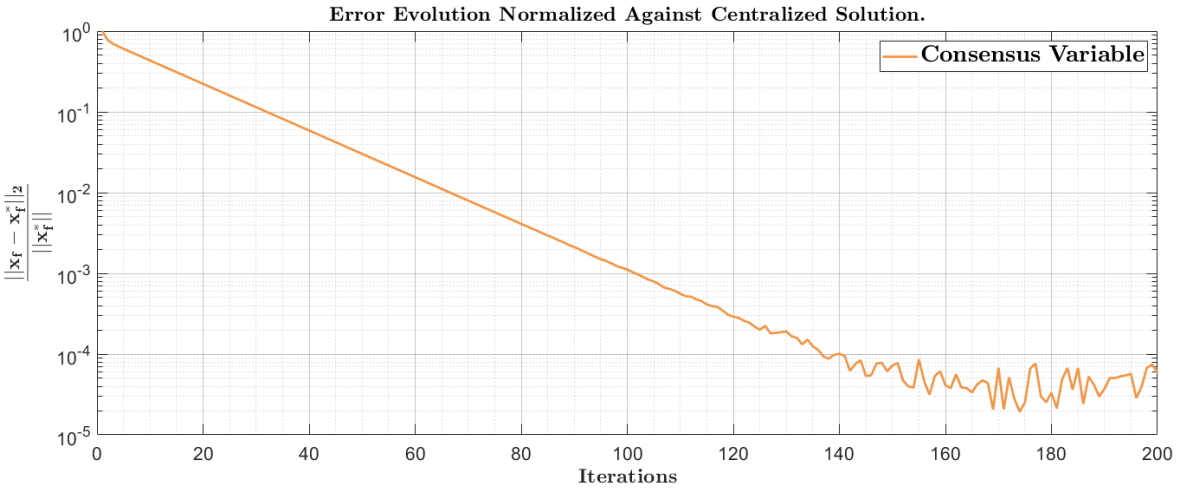

Figure 13: Error between consensus variable final states and the centralized solution.

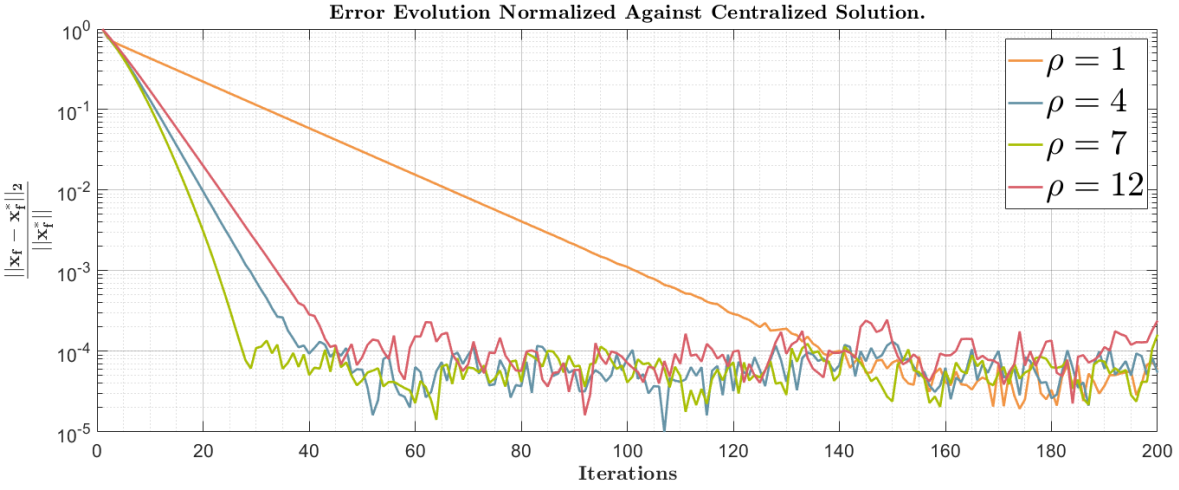

Figure 14: Error between consensus variable final states and the centralized solution.

improvements aren't linear, as penalty parameters $\rho > 7$ result in a decrease in the convergence rate of the algorithm. The example in Figure 14 is $\rho = 12$, which converges slower than $\rho = 7$.

To investigate this behaviour, a range of values $\rho \in [0.1, 100]$ was attempted, and the number of iterations needed to converge to within $\varepsilon = 10^{-}3$. Figure 15 shows the results. A fast decrease is visible in the range $\rho \in [0.1, 6.8]$. After that value, however, the number of iterations starts to increase again.

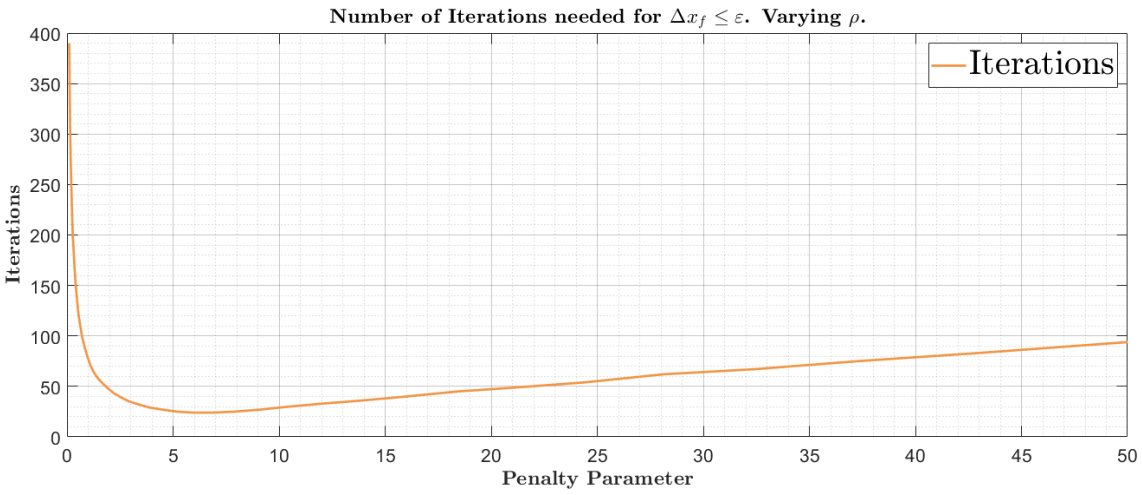

Figure 15: Iterations needed until $\Delta x_f^k \leq \varepsilon$ for varying penalty parameter values.

References

[1] Pontus Giselsson, Minh Dang Doan, Tamás Keviczky, Bart De Schutter, and Anders Rantzer. Accelerated gradient methods and dual decomposition in distributed model predictive control. *Automatica*, 49(3):829–833, 2013.

[2] Björn Johansson, Tamas Keviczky, Mikael Johansson, and Karl Henrik Johansson. Subgradient methods and consensus algorithms for solving convex optimization problems. In *2008 47th IEEE Conference on Decision and Control*, pages 4185–4190, 2008.

[3] Björn Johansson, Alberto Speranzon, Mikael Johansson, and Karl Henrik. Johansson. Distributed model predictive consensus. In *Proceedings of the 17th Symposium on Mathematical Theory of Networks and Systems*, page 2438–2444, 2006.

[4] Tamás Keviczky and Manuel Mazo Jr. Lecture slides on networked and distributed control systems [sc42100], 2021.

[5] Reza Olfati-Saber, J. Alex Fax, and Richard M. Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, 2007.