Assignment 4

SC42101 - Networked and Distributed Control

J.C.S. Amato (4688368)
Delft Center for Systems and Control

June 21, 2024

1 Multi-Aircraft Coordination

1.a Dual-Decomposition-Based Solution

1.a.1 Mathematical Derivation

We consider a finite-time optimal rendez-vous (FTOR) problem with 4 dynamical agents. Each agent $i \in \mathcal{I} := \{1,2,3,4\}$ is subject to dynamics described by:

$$x_i(t+1) = A_i x_i(t) + B_i u_i(t), \quad x_i(0) = x_{i,0} \tag{1.1}$$

Where the time-index set $t \in \mathcal{T} := \{0, \dots, T_{final} - 1\}$, $x \in \mathbb{R}^{n_x}$ and $u \in \mathbb{R}^{n_u}$.One can partition the set of agents in a set of master agents $\mathcal{A}_m := \{1\}$ and a set of slave agents $\mathcal{A}_s := \{2,3,4\}$ such that $\mathcal{I} = \{\mathcal{A}_s, \mathcal{A}_m\}$. A visualization of the above described situation is given in figure 1. The goal of this section is to derive a dual-decomposition-based algorithm that uses the projected subgradient method to solve the FTOR problem.

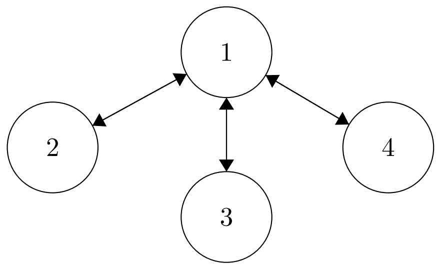

Figure 1: High-level visualization representing the information exchange between the dynamic agents

The dynamics of the complete finite horizon can be otbained by defining the dynamic system $\overline{X}_i = \Phi_i x_{0,i} + \Gamma_i \overline{u}_i$, where the matrices:

$$\Phi_i = \begin{pmatrix} A_i \\ A_i^2 \\ A_i^3 \\ \vdots \\ A_i^{T_f} \end{pmatrix}, \ \Gamma_i = \begin{pmatrix} B_i & 0 & \cdots & 0 \\ A_i B_i & B_i & \cdots & 0 \\ A_i^2 B_i & AB_i & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A_i^{T_f-1} B_i & A_i^{T_f-2} B_i & \cdots & B_i \end{pmatrix} \tag{1.2}$$

$$\overline{X}_i = \begin{pmatrix} x_i(1) & x_i(2) & \cdots & x_i(T_f) \end{pmatrix}^T \tag{1.3}$$

$$\overline{u}_i = \begin{pmatrix} u_i(0) & u_i(2) & \cdots & u_i(T_f - 1) \end{pmatrix} \tag{1.4}$$

Where $\Phi_i \in \mathbb{R}^{(n_x T_f) \times n_x}$, $\Gamma_i \in \mathbb{R}^{(n_x T_f) \times (n_u T_f)}$. With these definitions, we can rewrite the global objective function J in a more compact form for 4 agents:

$$\begin{aligned} J &= \sum_{i=1}^{4} \sum_{t=1}^{T_f} x_i(t)^T x_i(t) + u_i(t)^T u_i(t) \\ &= \sum_{i=1}^{4} \overline{X}_i^T \overline{X}_i + \overline{u}_i^T \overline{u}_i \\ &= \sum_{i=1}^{4} (\Phi_i x_{0,i} + \Gamma_i \overline{u}_i)^T (\Phi_i x_{0,i} + \Gamma_i \overline{u}_i) + \overline{u}_i^T \overline{u}_i \\ &= \sum_{i=1}^{4} x_{0,i}^T \Phi_i^T \Phi_i x_{0,i} + x_{0,i}^T \Phi_i^T \Gamma_i \overline{u}_i + \overline{u}_i^T \Gamma_i^T \Phi_i x_{0,i} \\ &\qquad + \overline{u}_i^T \Gamma_i^T \Gamma_i \overline{u}_i + \overline{u}_i^T \overline{u}_i \\ &= \sum_{i=1}^{4} x_{0,i}^T \Phi_i^T \Phi_i x_{0,i} + \underbrace{2 x_{0,i}^T \Phi_i^T \Gamma_i}_{c_i^T} \overline{u}_i + \overline{u}_i^T \underbrace{(\Gamma_i^T \Gamma_i + I)}_{H_i} \overline{u}_i \\ &= \sum_{i=1}^{4} \underbrace{\overline{u}_i^T H_i \overline{u}_i + c_i^T \overline{u}_i}_{f_i} = \sum_{i=1}^{4} f_i(\overline{u}_i), \end{aligned} \tag{1.5}$$

In the fifth equality we make use of the fact that $x_{0,i}^T \Phi_i^T \Phi_i x_{0,i}$ is a scalar and therefore symmetric. In addition, the constant term $x_{0,i}^T \Phi_i^T \Phi_i x_{0,i}$ does not affect the position of the optimizer and is thus removed in the sixth equality. We have arrived at a convenient form that allows us to define the the centralized optimization problem.

Definition 1.1 *Centralized Primal Problem. The FTOR problem can be stated in the following way:*

$$\begin{aligned} \min_{\overline{u}_i} & \sum_{i=1}^{4} f_i(\overline{u}_i) \\ \textit{subject to } & \overline{u}_i \in \mathbb{U}, \\ & \mathbf{h}_2(\overline{u}_1, \overline{u}_2) = 0 \\ & \mathbf{h}_3(\overline{u}_1, \overline{u}_3) = 0 \\ & \mathbf{h}_4(\overline{u}_1, \overline{u}_4) = 0 \end{aligned}$$

Where $\mathbf{h}_i(\overline{u}_1, \overline{u}_i) = A^{T_f} x_{o,1} + \Gamma_1^{T_f} \overline{u}_1 - A^{T_f} x_{o,i} - \Gamma_i^{T_f} \overline{u}_i, \quad \forall\, i \in \mathcal{A}_s$ *and the feasible set:*

$$\mathbb{U} := \{\overline{u}_i \in \mathbb{R}^{n_u(T_f-1)} \mid \begin{pmatrix} I & 0 \\ 0 & -I \end{pmatrix} \begin{pmatrix} \overline{u}_i \\ \overline{u}_i \end{pmatrix} \leq \begin{pmatrix} \mathbf{1}_{T_f-1} \frac{u_{max}}{T_f} \\ \mathbf{1}_{T_f-1} \frac{u_{max}}{T_f} \end{pmatrix} \}$$

The (complicating) coupling constraints ensure that the final positions and velocities of the aircraft are equal. Furthermore, $\Gamma_i^{T_f} \in \mathbb{R}^{n_x \times n_u T_f}$ denotes the bottom n_x rows of Γ_i corresponding to the effect of the input sequence on $x_i(T_f)$.

To implement the centralized primal problem in MATLAB it is necessary to reformulate definition 1.1 slightly differently. First, define the sequence $\overline{\mathbf{u}} = \begin{pmatrix} \overline{u}_i^T & \dots & \overline{u}_4^T \end{pmatrix}^T$ and the block diagonal matrix $\mathbf{H} = blkdiag(H_i, \dots, H_4)$. Furthermore, we can define the linear coefficient $\mathbf{c} = \begin{pmatrix} c_i^T & \dots & c_4^T \end{pmatrix}^T$. Then, we can state the centralized problem as:

$$\begin{aligned} \min_{\overline{\mathbf{u}}} \quad & \overline{\mathbf{u}}^T \mathbf{H} \overline{\mathbf{u}} + \mathbf{c}^T \overline{\mathbf{u}} \\ s.t. \quad & \overline{\mathbf{u}} \in \mathbb{U} \\ & \mathbf{A}_{eq} \overline{\mathbf{u}} = \mathbf{b}_{eq} \end{aligned} \tag{1.6}$$

The matrix $\mathbf{A}_{eq}$ is given by:

$$\mathbf{A}_{eq} = \begin{bmatrix} \Gamma_1^{T_f} & -\Gamma_2^{T_f} & 0 & 0 \\ \Gamma_1^{T_f} & 0 & -\Gamma_3^{T_f} & 0 \\ \Gamma_1^{T_f} & 0 & 0 & -\Gamma_4^{T_f} \end{bmatrix} \tag{1.7}$$

The vector $\mathbf{b}_{eq}$ is given by:

$$\mathbf{b}_{eq} = \begin{bmatrix} -A_1^{T_f} x_{0,1} + A_2^{T_f} x_{0,2} \\ -A_1^{T_f} x_{0,1} + A_3^{T_f} x_{0,3} \\ -A_1^{T_f} x_{0,1} + A_4^{T_f} x_{0,4} \end{bmatrix} \tag{1.8}$$

Subsequently, we aim to perform dual decomposition on the formulated problem. Thus, we introduce the Lagrange multiplier $\lambda \in \mathbb{R}^{12}$ partitioned as $\lambda := \begin{pmatrix} \lambda_{1,2} & \lambda_{1,3} & \lambda_{1,4} \end{pmatrix}^T$. Each "sub-multiplier" corresponds to the coupling between the master and the slave agent. We can then define the Lagrangian:

$$L(\overline{u}_i, \lambda) = \sum_{i=1}^{4} f_i(\overline{u}_i) + \lambda^T \begin{pmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \mathbf{h}_3 \end{pmatrix} \tag{1.9}$$

Which can be separated into:

$$\begin{aligned} L_1(\overline{u}_1, \lambda) = f_1(\overline{u}_1) &+ \lambda_{1,2}^T (A^{T_f} x_{o,1} + \Gamma_1^{T_f} \overline{u}_1) \\ &+ \lambda_{1,3}^T (A^{T_f} x_{o,1} + \Gamma_1^{T_f} \overline{u}_1) \\ &+ \lambda_{1,4}^T (A^{T_f} x_{o,1} + \Gamma_1^{T_f} \overline{u}_1) \end{aligned} \tag{1.10}$$

$$L_i(\overline{u}_i, \lambda) = f_i(\overline{u}_i) - \lambda_{1,i}^T (A^{T_f} x_{o,i} + \Gamma_i^{T_f} \overline{u}_i) \tag{1.11}$$

Leading to separable dual functions:

$$d_i(\lambda) = \inf_{\overline{u}_i} L_i(\overline{u}_i, \lambda_{1,i}) \tag{1.12}$$

We can thus proceed to define the dual subproblems in a compact manner. Call $e_j^N \in \mathbb{R}^N$ the j_{th} unit vector such that $e_1^N = \begin{pmatrix} 1 & 0 & \dots & 0 \end{pmatrix}^T$, $e_2^N = \begin{pmatrix} 0 & 1 & \dots & 0 \end{pmatrix}^T$ and the identity matrix $I_N \in \mathbb{R}^N$ is noted as:

$$I_N = \begin{pmatrix} e_1^N & e_2^N & \dots e_N^N \end{pmatrix} \tag{1.13}$$

Definition 1.2 *Dual Subproblem. At the lower level the dual subproblem of the FOTR for the master node is defined as:*

$$\begin{aligned} \min_{\overline{u}_1} \quad & f_1(\overline{u}_1) + \lambda^T (\Gamma_1^{T_f} \overline{u}_1 \otimes \mathbf{1}_3) \\ s.t.: \quad & \overline{u}_1 \in \mathbb{U} \end{aligned} \tag{1.14}$$

For the slave nodes $i \in \mathcal{A}_s$ the subproblems are defined as:

$$\begin{aligned} \min_{\overline{u}_i} \quad & f_i(\overline{u}_i) - \lambda^T \begin{pmatrix} e_p^r & \dots & e_{p+n_x-1}^r \end{pmatrix} \Gamma_i^{T_f} \overline{u}_i \\ s.t.: \quad & \overline{u}_i \in \mathbb{U} \end{aligned} \tag{1.15}$$

Where $p = (i-1)n_x + 1$ and $r = 12$, leading to dual functions $d_i(\lambda)$.

Note that the constant terms have been removed again as they do not affect the position of the optimizer. Then we can define the dual master problem.

Definition 1.3 *Dual Master Problem. At the higher level, the dual master problem updating the dual variable $\lambda \in \mathbb{R}^r$ is defined as:*

$$\begin{aligned} \max_{\lambda} \quad & \sum_{i}^{4} d_i(\lambda) - \lambda^T K \\ s.t.: \quad & \lambda \in \mathbb{R}^+ \end{aligned} \tag{1.16}$$

Where

$$K = \begin{pmatrix} A_2^{T_f} x_{0,2} - A_1^{T_f} x_{0,1} \\ A_3^{T_f} x_{0,3} - A_1^{T_f} x_{0,1} \\ A_4^{T_f} x_{0,4} - A_1^{T_f} x_{0,1} \end{pmatrix}$$

The constant term $A_i^{T_f} x_{0,i}$ that is removed in the subproblems can be found in the dual master problem. To find the solution of the latter, we perform subgradient iterations on the dual variable:

$$\lambda_{k+1} = \lambda_k + \alpha_k \begin{pmatrix} x_1^{(k)}(T_f) - x_2^{(k)}(T_f) \\ x_1^{(k)}(T_f) - x_3^{(k)}(T_f) \\ x_1^{(k)}(T_f) - x_4^{(k)}(T_f) \end{pmatrix} \tag{1.17}$$

Here $x_i^{(k)}$ denotes the state at final time T_f found by application of the optimal input-trajectory at iteration k. α_k denotes the step-size at iteration k, assumed as non-constant for generality. The error sequence has been chosen as:

$$\epsilon = \frac{||\overline{\mathbf{u}}^{(k)} - \overline{\mathbf{u}}^*||}{||\overline{\mathbf{u}}^*||} \tag{1.18}$$

Where $\overline{\mathbf{u}}^{(k)}$ and $\overline{\mathbf{u}}^*$ denote the optimal sequences (see the formulation of the centralized problem) found at iteration k and by the centralized solution, respectively.

1.A.2 Numerical Simulations

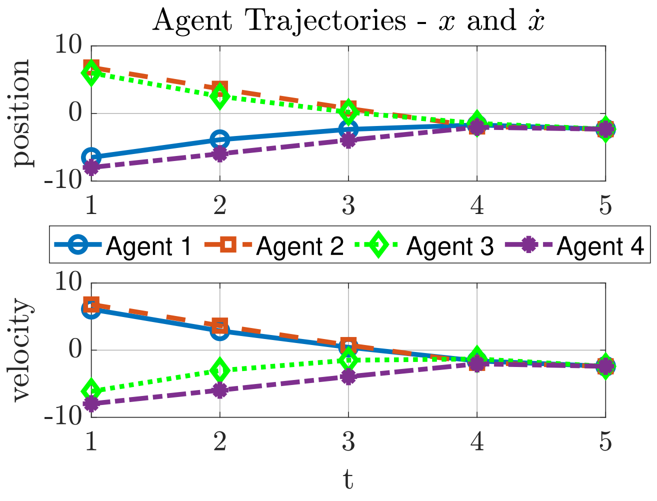

Figure 2: Agent trajectories in the x space after convergence of the centralized solution

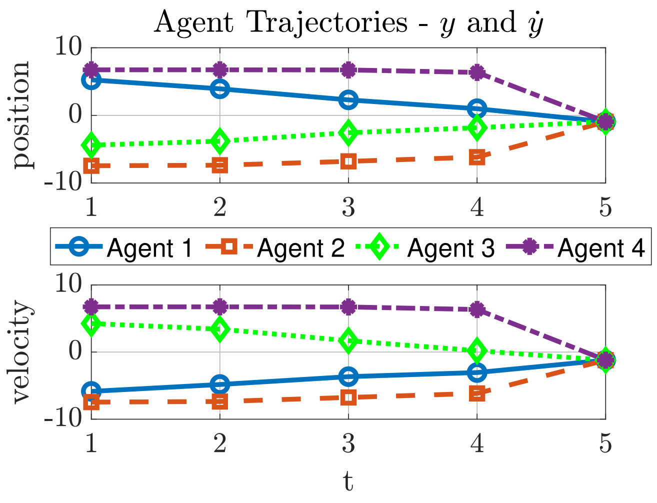

Figure 3: Agent trajectories in the y space after convergence of the centralized solution

We can proceed to analyze the convergence properties of the algorithm by comparing the results with those of the centralized solution. Hence, let us firstly observe the agent trajectories after convergence of the latter, which produces an optimizer $\overline{\mathbf{u}}^*$. The trajectories in the x space are shown in figure 2 whilst those in the y space are shown in figure 3. It can be seen that in the x space the aircrafts converge rather gradually whilst in the y space we observe a more sudden step at $t = 4$.

We can compare the trajectories of the centralized and decentralized (dual) solution visually by observing figures 4 and 5. It is clear that there is very little difference, hardly noticeable by the blind eye. Figure 1.18 shows the convergence of the error sequence defined earlier with a constant step size $\alpha = 0.5$. The algorithm converges to a steady value in 622 iterations, reaching an accuracy of approximately $\epsilon = 1E - 10$.

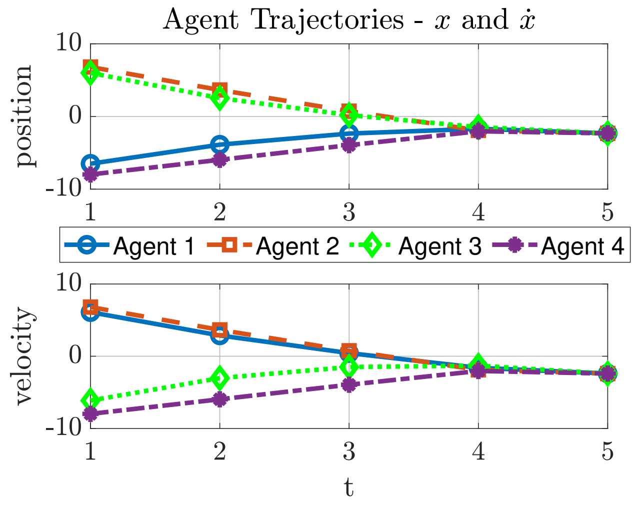

Figure 4: Agent trajectories in the x space after convergence of the decentralized solution with the dual subgradient method

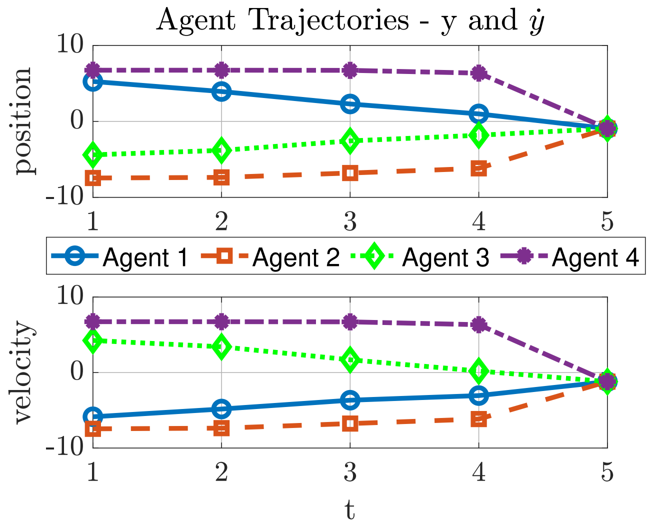

Figure 5: Agent trajectories in the y space after convergence of the decentralized solution with the dual subgradient method

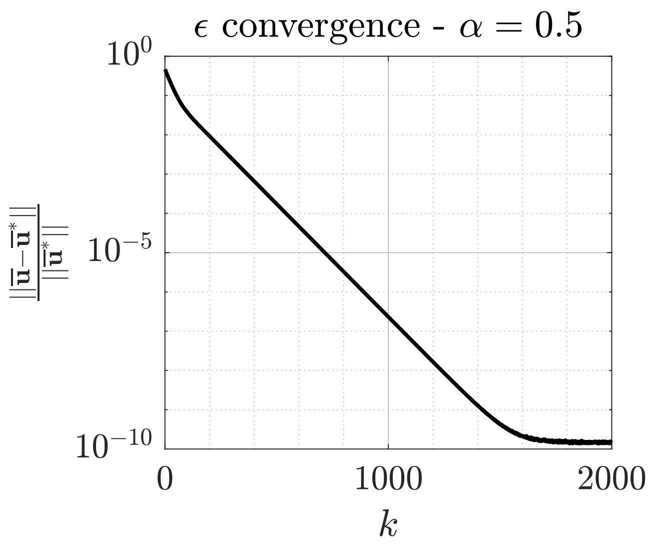

Figure 6: Convergence of the error sequence ϵ (see equation 1.18) with a constant step size $\alpha = 0.5$

1.B Step Size and -Sequence

Here, we first investigate the effect of the step size (with a constant step) on the convergence properties of the dual subgradient algorithm. Secondly, we show what happens when choosing some specific step size update sequences.

1.B.1 Effect of Step Size (Constant)

In figure 7 the error sequence is shown for different but constant step sizes. It is clear that when using the highest step size, we obtain the fastest convergence. The asymptote to which the algorithm converges doesn't seem to vary with step size and is approximately $\epsilon = 1e-9$. However, when normalizing the subgradient, the result in figure 8 shows that there is a speed-accuracy trade-off. That is, with a lower constant step-size, the convergence is slower but comes closer to the optimum.

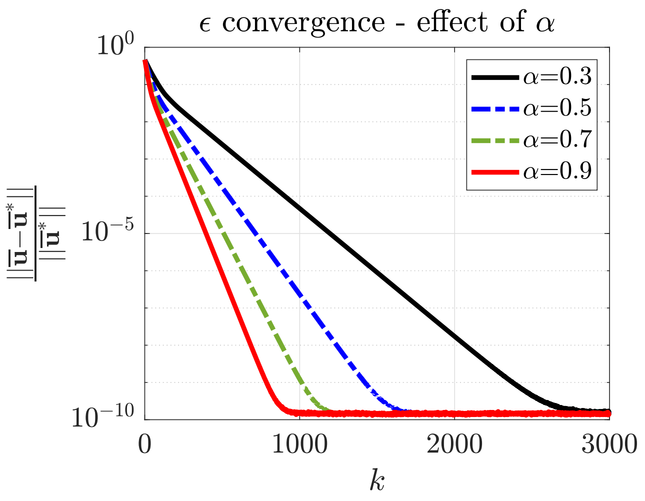

Figure 7: Convergence of the error sequence ϵ for different values of a constant step size

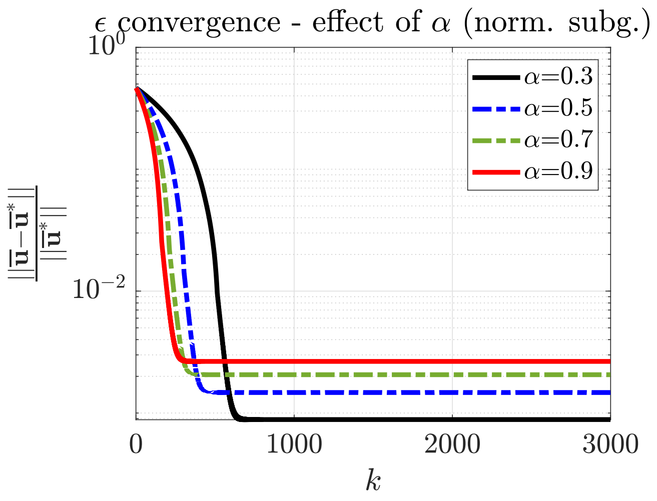

Figure 8: Convergence of the error sequence ϵ for different values of a constant step size and a normalized subgradient

1.B.2 Diminishing Step Size: Method I

The first method for diminishing step size is given by the equation:

$$\alpha_k = \frac{a}{b+k} \tag{1.19}$$

Where $a > 0$ and $b \geq 0$. The method is obtained from reference [1] and fulfills the following conditions:

$$\alpha_k \geq 0, \quad \sum_{k=1}^{\infty} a_k^2 < \infty, \quad \sum_{k=1}^{\infty} a_k = \infty \tag{1.20}$$

That is, square summable but not summable.

By experimenting with both b and a it was clear b should be chosen quite small, otherwise the error would converge quickly to a relatively large value. Furthermore, the effect of choosing a different b was not so large. Hence in figure 9 the effect of a on convergence is shown with a constant $b = 1$.

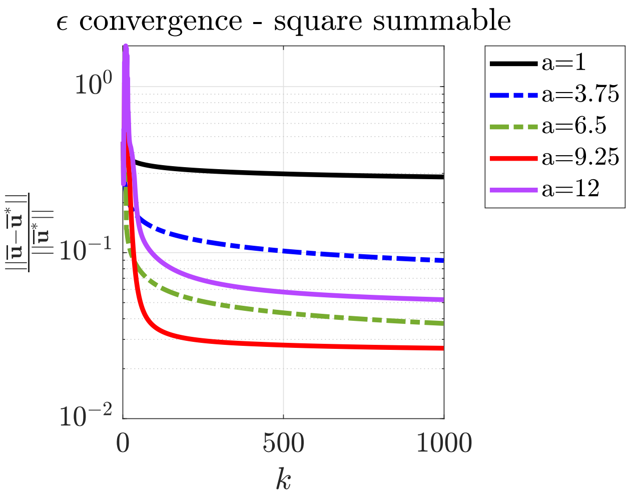

Figure 9: Convergence of the error sequence ϵ for different values of a using step the size update from equation 1.19

Compared to a constant step size, the result is not so good in terms of residual, as it is many orders of magnitude higher after convergence. In that respect, it seems like there is some kind of "sweet spot" for parameter a. Observe that the residual decreases until $a = 9.25$, but when using $a = 12$ it is higher.

However, convergence happens much faster compared to a constant step size. When using a somewhat lower value like $a = 1$, the method converges after about 100 iterations, albeit with a large residual.

In conclusion, we have shown two trade-offs that have to be taken into account, if one would us this step-size update. Firstly, one has to carefully choose a to minimize the residual. Secondly, there is a speed-accuracy trade-off in the choice between constant step size and *Method I*, in that order.

1.B.3 Diminishing Step Size: Method II

The second method is similar, although it possesses fairly different properties. The update equation is given by:

$$\alpha_k = \frac{\gamma}{\sqrt{k}} \tag{1.21}$$

where $\gamma > 0$. The method is obtained form reference [1], and is non-summable:

$$\alpha_k \geq 0, \quad \lim_{k\to\infty} \alpha_k = 0, \quad \sum_{k=1}^{\infty} \alpha_k = \infty \tag{1.22}$$

In figure 10 the convergence for various values of γ is shown. In comparison to *Method I*, it is clear that a much lower error can be reached, albeit in more iterations. A striking observation can be made about the choice of γ. When choosing $\gamma = 5$ or $\gamma = 4$ the method converges very quickly to a value, temporary stabilizes, and continues to converge after some time.

In conclusion, this step-size update sequence is better than *Method 2* in terms of residual error. Compared to a constant step size, the residual error reached in an equal number of iterations is higher, hence in that respect it is not an attractive option.

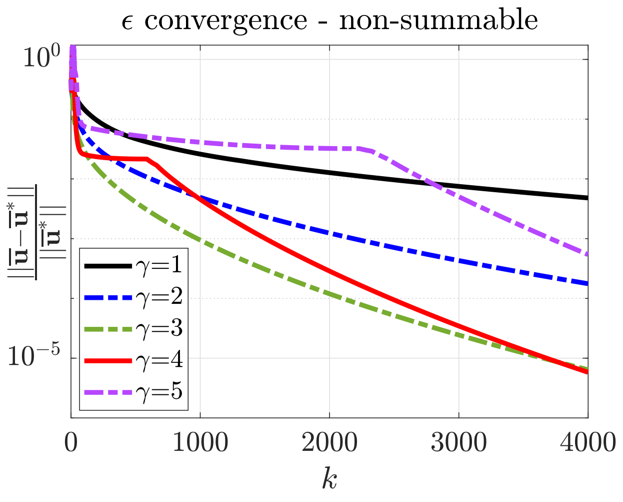

Figure 10: Convergence of the error sequence ϵ for different values of γ using step the size update from equation 1.21

1.C Accelerated Subgradient Method

We can attempt to speed up convergence rates by implementing an accelerated subgradient method. The method used here is what Polyak calls the *Heavy Ball Method*. It consists of the implementing a second-order term into the dynamical system, that is the dual variable update. Hence the term heavy ball, which alludes to taking into account 'momentum' as the higher-order term representing the past update direction that was taken. The update equation in our case is given by:

$$\begin{aligned}\lambda_{k+1} = \lambda_k + \alpha_k &\begin{pmatrix} x_1^{(k)}(T_f) - x_2^{(k)}(T_f) \\ x_1^{(k)}(T_f) - x_3^{(k)}(T_f) \\ x_1^{(k)}(T_f) - x_4^{(k)}(T_f) \end{pmatrix} \\ &+ \beta_k(\lambda_k - \lambda_{k-1})\end{aligned} \tag{1.23}$$

By tuning β we can tune the effect of past update directions on the current update. At first, we use a constant $\alpha = 0.5$ and vary β between 0.1 and 0.9, which is shown in figure 11.

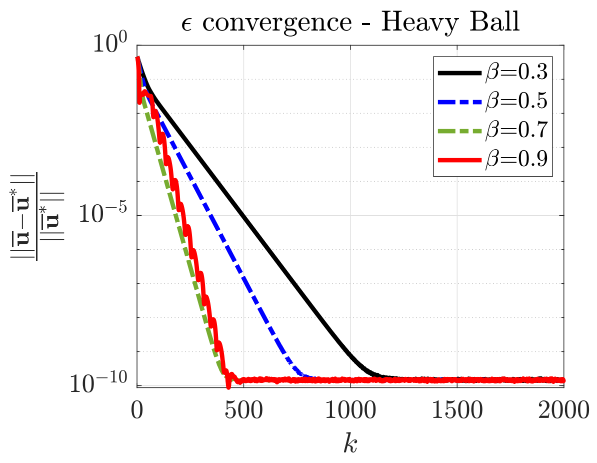

Figure 11: Convergence of the error sequence ϵ for different values of β using the heavy ball method and a constant $\alpha = 0.5$

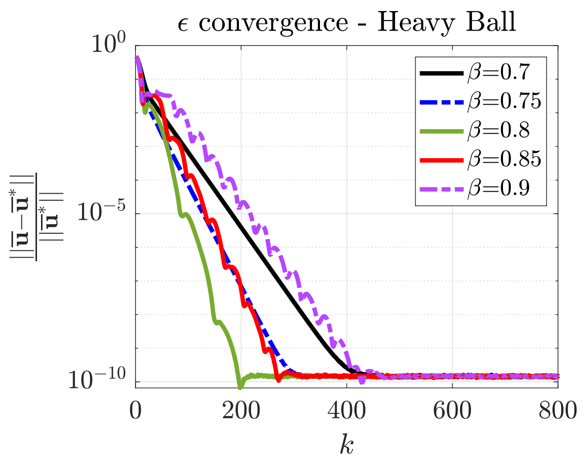

Figure 12: Convergence of the error sequence ϵ for different values of β using the heavy ball method and a constant $\alpha = 0.5$

It is clear that using $\beta > 0$ speeds up convergence in comparison to a constant step-size only (see Figure 7 and note the different horizontal axis). A value closer to 1 makes for an interesting effect with an oscillating error sequence. Let us investigate closer the values near to 1, shown in Figure 12. Again, there seems to be a "sweet spot" for the value of β, around $\beta = 0.8$. Thereafter, adding more weight to the momentum term only increases the time of convergence and adds many oscillations, which could be harmful when choosing certain stopping criteria.

1.D Combined Consensus/Incremental Subgradient

The combined consensus/incremental subgradient method is inspired from references [2, 3]. The problem at hand must be reformulated slightly. Earlier, we were directly coupling the final states of each agent. Now, each agent will have a local version of the final state estimate θ_i. The aim is to ensure the local estimates converge after a finite number of iterations. In addition, we will investigate how the solution compares to the centralized solution and the dual-decomposition based solution.

1.D.1 Mathematical Formulation

Consider the local parametric optimization problems:

$$\begin{aligned} q_i(\theta_i) = \min_{\overline{u}_i} \; & f_i(\overline{u}_i) \\ s.t. \; & A^{T_f} x_{o,i} + \Gamma_i^{T_f} \overline{u}_i = \theta_i \\ & \overline{u}_i \in \mathbb{U}, \theta_i \in \Theta \end{aligned} \tag{1.24}$$

That is, optimize the local cost function with decision variable $\overline{u}_i$, given a value for θ_i. The master problem is then given by:

$$\begin{aligned} \min_{\theta_i} \; & \sum_{i=1}^{4} q_i(\theta_i) \\ s.t. \; & \theta_i \in \Theta_i, \; \forall i \in \mathcal{A} \end{aligned} \tag{1.25}$$

The set Θ_i is the reachable set after time T_f using the set of feasible inputs.

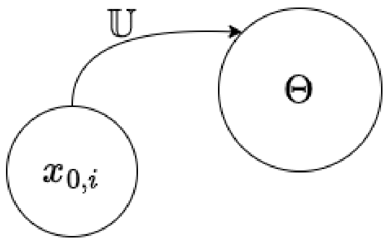

Figure 13: Visualization of the reachable set

For each agent, we can define by Lagrangian relaxation:

$$L_{i\in\mathcal{A}}(\overline{u}_i,\lambda_i,\theta_i) = f_i(\overline{u}_i) + \lambda_i^T(A^{T_f}x_{o,i} + \Gamma_i^{T_f}\overline{u}_i - \theta_i) \tag{1.26}$$

With the local Lagrange multipliers λ_i. Then we obtain the dual functions, where we remove the constant terms:

$$d_{i\in\mathcal{A}}(\lambda_i,\theta_i) = \inf_{\overline{u}_i} L(\overline{u}_i,\lambda_i,\theta_i) = f_i(\overline{u}_i) + \lambda_i^T\Gamma_i^{T_f}\overline{u}_i \tag{1.27}$$

Subsequently, the steps taken by the algorithm are explained. First, at each iteration the optimal Lagrange multipliers are found by:

$$\lambda_i^k := \arg\sup_{\lambda} d_{i\in\mathcal{A}}(\lambda_i,\theta_i) \tag{1.28}$$

In Matlab this is achieved by solving the Lagrangian as a quadratic programming problem with `quadprog()`, from which the dual variables can be directly extracted.

The dynamical agents then perform an update procedure of their local rendez-vous point estimate in parallel. First, they exchange information with their neighbors for ϕ number of consensus iterations. Then, the agents implement a local component update. The procedure is given by:

$$\theta_i^{k+1} = \mathcal{P}_\Theta\left[\sum_{j=1}^{N}[W^\varphi]_{ij}\left(\theta_k^j + \alpha_k\lambda_j^k(\theta_j^k)\right)\right] \tag{1.29}$$

It can be shown that λ_i^k is a subgradient of $\sup_\lambda d_{i\in\mathcal{A}}(\lambda_i,\cdot)$ at θ_i (see [3], equivalent to the proof in assignment 3).

It is important to note that the θ_i is (possibly) projected on the reachable set at each iteration. Therefore, the method to verify whether the next iterate θ_i^{k+1} is in the reachable set and the projection method are explained.

The input set $\mathbb{U}$ is a hyperrectangular set, with 4 vertices $\mathcal{U} := \{\overline{u}_i^{\overline{m},\overline{m}}, \overline{u}_i^{\underline{m},\overline{m}}, \overline{u}_i^{\overline{m},\underline{m}}, \overline{u}_i^{\underline{m},\underline{m}}\} := \{\mathcal{U}_1,\mathcal{U}_2,\mathcal{U}_2,\mathcal{U}_4\}$. Here $\overline{u}_i^{\overline{m},\overline{m}}$ denotes the maximum input value on the first and second input during the complete input sequence. Similarly, we can denote the set of terminal states obtained by applying these input trajectories for each agent as $\mathcal{X}_i^f := \{x_i^f(\mathcal{U}_1),\ldots,x_i^f(\mathcal{U}_4)\}$. By linearity of the system, the reachable set is given by the affine hull of the elements of $\mathcal{X}_i^f$:

$$\Theta_i = \mathrm{aff}(\mathcal{X}_i^f) = \{\sum_{j=1}^{4}\mu_j x_i^f(\mathcal{U}_j) \mid \mu_i\in\mathbb{R},\ \sum_{i=1}^{4}\mu_i = 1\} \tag{1.30}$$

We can therefore verify at every iteration if the computed $\hat{\theta}_i$ is in the set Θ_i by verifying the existence of μ_j. More precisely, let $M := \begin{pmatrix} x_i^f(\mathcal{U}_1) & \ldots & x_i^f(\mathcal{U}_4)\end{pmatrix}$, then $\hat{\theta}_i\in\Theta_i$ if there exist μ_i such that the following equation is solvable:

$$\begin{pmatrix} M \\ \mathbf{1}_4^T \end{pmatrix}\mu_i = \begin{pmatrix}\hat{\theta}_i \\ 1\end{pmatrix} \tag{1.31}$$

Which can be easily solved in Matlab as a linear program. If $\hat{\theta}_i\in\Theta_i$ we set $\theta_{k+1} = \hat{\theta}_i$, otherwise we project onto the convex hull with the projection matrix:

$$\hat{\theta}_i = M(M^TM)^\dagger M^T \tag{1.32}$$

Where the super script $\dagger$ denotes the Moore-Penrose pseudo-inverse.

1.d.2 Numerical Simulations

The algorithm has been run for different values of ϕ to observe the effect of the number of consensus iterations on convergence. The step-size update has been chosen as:

$$\alpha_k = \frac{0.05}{\sqrt{(1E3k)}} \tag{1.33}$$

The exact values for the step-size update have been chosen by iteration.

Application of the algorithm shows convergence, albeit not completely towards the optimum of the centralized solution. Figures 14 and 15 show the aircraft trajectories after convergence using $\phi = 20$. It can be seen that at the final time there is still some offset, especially in the x space for agent 4. It is therefore interesting to again observe the error sequence.

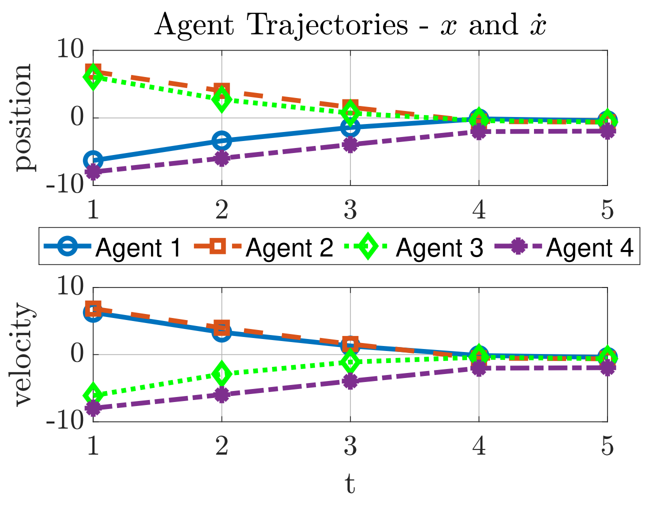

Figure 14: Convergece of the aircraft trajectories in the x space using the combined consensus/incremental subgradient method

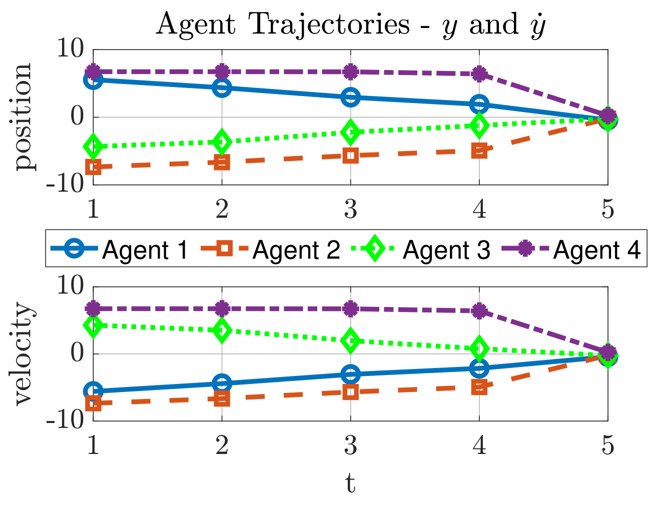

Figure 15: Convergece of the aircraft trajectories in the y space using the combined consensus/incremental subgradient method

Figure 16 shows the error convergence using the projected consensus/incremental subgradient update. The trend of the convergence is as expected, but the error is higher than expected.

Indeed, as in reference [2] the error sequence converges quickly. Furthermore, convergence is faster and more precise when using a higher number of consensus iterations. However, the absolute value of the error is quite high in comparison to the other methods. In addition, the difference in result between different numbers of consensus iterations is quite small.

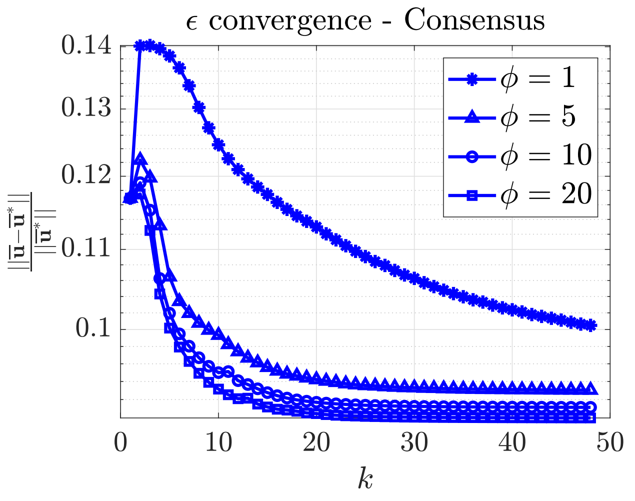

Figure 16: Convergence of the error sequence using the combined consensus/incremental subgradient method

To improve the convergence properties I have implemented the heavy ball method, which is shown in figure 17. In addition, I have introduced some noise to possibly slow down convergence but increasing the chance of escaping local optima, this is shown in figure 18. Unfortunately, the convergence properties have clearly not improved significantly.

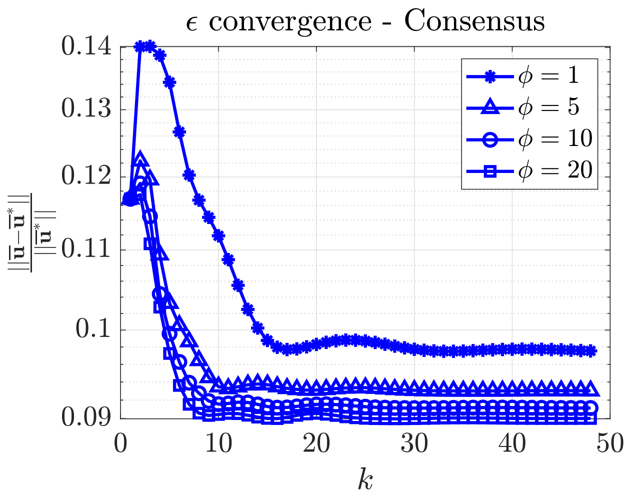

Figure 17: Convergence of the error sequence using the combined consensus/incremental subgradient method and the heavy ball acceleration method

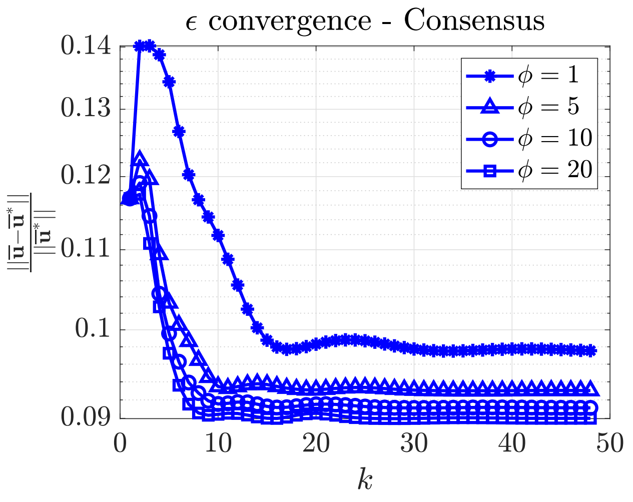

Figure 18: Convergence of the error sequence using the combined consensus/incremental subgradient method, the heavy ball acceleration method and noise

2 ADMM with Consensus

The goal of this section is to use the Alternating Directions Method of Multipliers (ADMM) with consensus to solve the FTOR problem. In addition, we investigate its convergence towards the centralized solution and the effect of the regularization parameter ρ.

We consider the problem of the form:

$$\begin{aligned} &\min_{\overline{u}_i} \sum_{i=1}^{4} f_i(\overline{u}_i) \\ &s.t.\ \overline{u}_i \in \mathbb{U} \\ &\quad x_i^f - z = 0 \\ &\quad \forall\ i \in \mathcal{A} \end{aligned} \tag{2.1}$$

Where z is a common (i.e., global) variable giving rise to the consensus contraint $x_i^f - z = A^{T_f}x_{0,1} + \Gamma_i^{T_f}\overline{u}_i - z = 0$. This encapsulates the goal of trajectory convergence of the dynamic agents at a common coordinate.

The ADMM is based on the augmented Lagrangian:

$$\begin{aligned} &L_\rho(x,z,\lambda) := \\ &:= \sum_{i=1}^{N}\left(f_i(\overline{u}_i) + \lambda_i^T(x_i^f - z) + \frac{\rho}{2}\|x_i^f - z\|_2^2\right) \end{aligned} \tag{2.2}$$

And on the sequential minimization of the local and global variables:

$$\begin{aligned} &\overline{u}_i^{k+1} := \\ &:= \arg\min_{\overline{u}_i}\left(f_i(\overline{u}_i) + (\lambda_i^k)^T(x_i^f - z^k) + \frac{\rho}{2}\|x_i^f - z^k\|_2^2\right) \end{aligned} \tag{2.3}$$

$$z^{k+1} := \frac{1}{4}\sum_{i=1}^{4}\left(x_i^{f,k} + \frac{1}{\rho}\lambda_i^k\right) \tag{2.4}$$

$$\lambda_i^{k+1} := \lambda_i^k + \rho(x_i^{f,k} - z^{k+1}) \tag{2.5}$$

Where λ_i^k denotes the local Lagrange multiplier and $x_i^{f,k}$ denotes the final state given by the input trajectory found in equation 2.3, both at iteration k. As we're optimizing over $\overline{u}_i$ it is necessary to remove the constant terms in the minimization problem. Let us work out the squared Euclidean norm:

$$\begin{aligned} &\|x_i^f - z^k\|_2^2 = \\ &= (A^{T_f}x_{0,1} + \Gamma_i^{T_f}\overline{u}_i - z)^T(A^{T_f}x_{0,1} + \Gamma_i^{T_f}\overline{u}_i - z) \\ &= \overline{u}_i^T(\Gamma_i^{T_f})^T\Gamma_i^{T_f}\overline{u}_i + 2x_{0,i}^TA^{T_f}\Gamma_i^{T_f}\overline{u}_i - 2z^T\Gamma_i^{T_f}\overline{u}_i \\ &\quad + x_{0,i}^T(A^{T_f})^TA^{T_f}x_{0,i} - 2x_{0,i}^TA^{T_f}z + z^Tz \end{aligned} \tag{2.6}$$

We therefore arrive at the definition of the minimization problem of the augmented Lagrangian.

Definition 2.1 *The minimization problem in the ADMM consensus algorithm to iteratively find a in input sequence in the fashion of equation 2.3 is given by:*

$$\begin{aligned} &\min_{\overline{u}_i} \overline{u}_i^T\mathcal{H}_i\overline{u}_i + \tilde{c}_i^T\overline{u}_i \\ &s.t.\ \overline{u}_i \in \mathbb{U} \end{aligned}$$

The matrix and vector in the cost function are given by:

$$\begin{aligned} \mathcal{H}_i &:= H_i + \frac{\rho}{2}(\Gamma_i^{T_f})^T\Gamma_i^{T_f} \\ \tilde{c}_i &:= c_i + \lambda_i^T\Gamma_i^{T_f} + \rho(A^{T_f}x_{0,i} - z)^T\Gamma_i^{T_f} \end{aligned}$$

Where the vector c_i and the matrix H_i are defined in equation 1.5.

Note that the scaling factors in equation 2.2 and 2.6 cancel out.

Figures 19 and 20 show the agent trajectories after convergence.

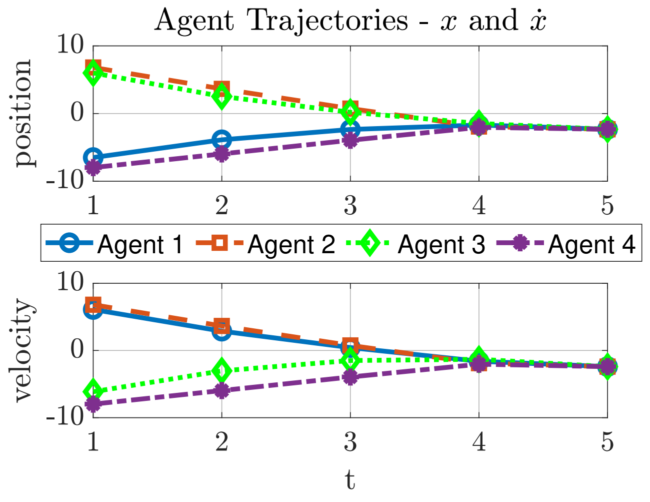

Figure 19: Agent trajectories in the x space after convergence of the ADMM algorithm

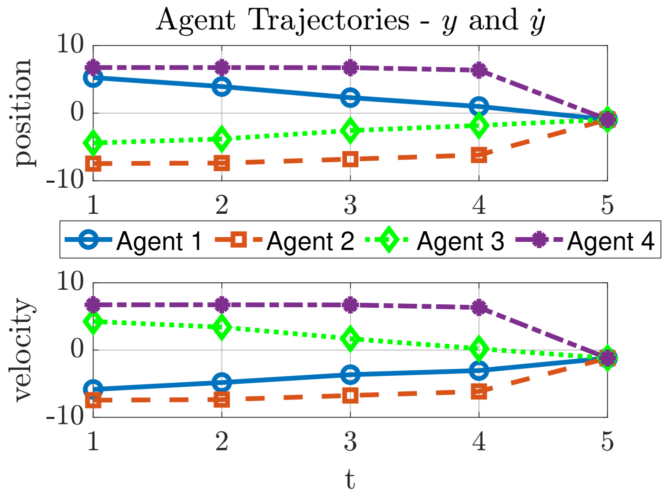

Figure 20: Agent trajectories in the x space after convergence of the ADMM algorithm

The aircraft trajectories after convergence are not different than the ones in the centralized solution. It is clear that the agents find a common rendez-vous point.

2..1 Effect of ρ on the convergence rate

In Figures 21 and 22 the effect of ρ on convergence rate of the algorithm is shown. As we've seen earlier, tuning a parameter is about finding the "sweet spot" for your problem. Indeed, when increasing ρ the convergence rate increases. However. too much increase may result in a degradation of the improved convergence rate. To investigate this further, figure 22 shows the relation between ρ and the number of iterations necessary to obtain at least an accuracy of $\epsilon_{min} = 1E-3$. According to the experiment, there is an optimum at about $\rho = 8$.

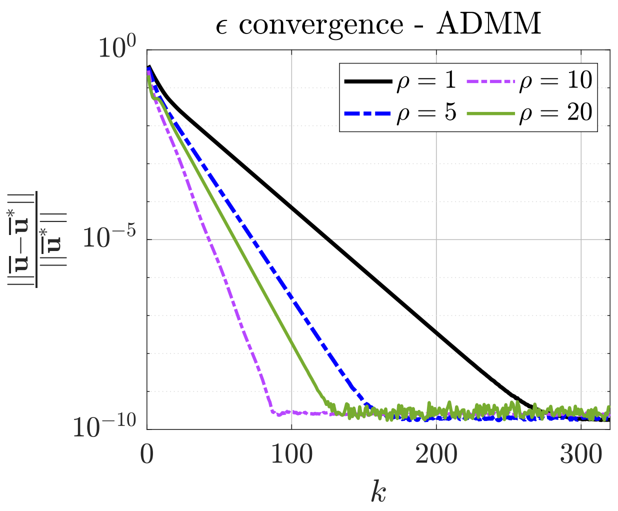

Figure 21: Error convergence of the ADMM method for different values of ρ

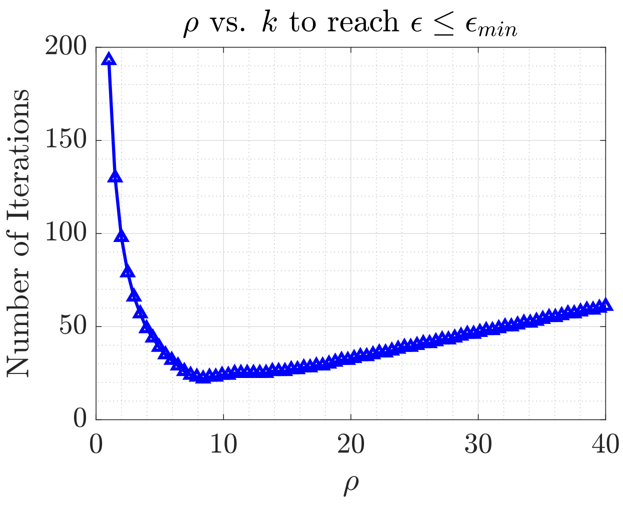

Figure 22: Number of iterations needed per value of ρ to reach an error threshold of $\epsilon \leq 0.001$

References

[1] Stephen Boyd. *Subgradient Method Notes.* `https://stanford.edu/class/ee364b/lectures/subgrad_method_notes.pdf`. Accessed: 2024-06-14. 2024.

[2] Bjorn Johansson et al. “Subgradient methods and consensus algorithms for solving convex optimization problems”. In: *2008 47th IEEE Conference on Decision and Control*. IEEE. 2008, pp. 4185–4190.

[3] Björn Johansson et al. “On decentralized negotiation of optimal consensus”. In: *Automatica* 44.4 (2008), pp. 1175–1179. ISSN: 0005-1098. DOI: `10.1016/j.automatica.2007.09.003`. URL: `https://www.diva-portal.org/smash/record.jsf?pid=diva2:335515`.