Assignment 4

SC42101: Networked and Distributed Control Systems

Richard Adam

Delft University of Technology

TU Delft

Introduction

This report investigates several methods of solving a multi-agent coordination problem in which each agent is represented by a Linear Time-Invariant system described by:

$$x_i(t+1) = A_i x_i(t) + B u_i(t), \quad i \in (1, ..., N)$$

Where N is the number of agents in the problem. The goal is a rendezvous problem such that for some time T_f, $x_i(T_f) = x_f, \ \forall \ i$. In addition, there is a limit applied to the control input such that for a given u_{max}, $|u_i(t)| \leq u_{max} \quad \forall \, i, t$. The objective function of the rendezvous is given as the quadratic function:

$$\sum_{i=1}^{N} \sum_{t=0}^{T_f - 1} [x_i(t)^T x_i(t) + u_i(t)^T u_i(t)] + x_i(T_f)^T x_i(T_f) \tag{1}$$

In the particular problem examined, there are four agents, each of with the same A_i and B_i matrices:

$$A_i = \begin{bmatrix} 0.626 & -0.6975 \\ 0.8719 & 0.626 \end{bmatrix}, \quad B_i = \begin{bmatrix} -0.0748 \\ 0.1744 \end{bmatrix} \quad \forall i \in (1, 2, 3, 4)$$

The agents also have starting conditions $x_{0,i}$, which for each agent are:

$$x_{0,1} = \begin{bmatrix} -10 \\ -0.5 \end{bmatrix}, x_{0,2} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, x_{0,3} = \begin{bmatrix} 2 \\ 0 \end{bmatrix}, x_{0,4} = \begin{bmatrix} 5 \\ -0.5 \end{bmatrix}$$

Hereinafter, when discussing the trajectories, the x_i varaible will be referred to as if it encodes an (x,y) position in the form $[x, y]^T$.

Question 1: Dual Decomposition

Each of the following solutions depend upon the dual decomposition method, whereby the so-called primal minimization problem is converted to a dual maximization problem and then decomposed agent-by-agent. The first step, of course, is to form the centralized primal problem. The objective function for the centralized primal problem is given in Equation 1. One can express the system dynamics as a constraint, formulated as:

$$x_i(t+1) = A_i x_i(t) + B_i u_i(t) \quad \forall i, t$$

The constraint on maximum control effort can be expressed as:

$$|u_i(t)| \leq u_{max} \quad \forall \, i, t$$

The final constraint is that all final positions must be the same, some x_f to be optimized as a parameter. The position x_i at any given t can be expressed as:

$$x(t) = A_i^t x_{0,i} + \sum_{k=0}^{t-1} A_i^{t-1-k} B_i u_i(k)$$

Thus, the constraint on final position can be expressed as:

$$A_i^{T_f} x_{0,i} + \sum_{k=0}^{T_f-1} A_i^{T_f-1-k} B_i u_i(k) - x_f = 0 \quad \forall i$$

To summarize, the problem can be expressed in its complete form as:

$$\begin{aligned} \min_{x_f, u_i} \sum_{i=1}^{N} \sum_{t=0}^{T_f-1} [x_i(t)^T x_i(t) + u_i(t)^T u_i(t)] + x_i(T_f)^T x_i(T_f) \\ \text{subject to:} \quad x_i(t+1) = A_i x_i(t) + B_i u_i(t) \quad \forall i, t \\ |u_i(t)| \leq u_{max} \quad \forall\, i, t \\ A_i^{T_f} x_{0,i} + \sum_{k=0}^{T_f-1} A_i^{T_f-1-k} B_i u_i(k) - x_f = 0 \quad \forall i \end{aligned}$$

It is clear that the third constraint is the coupling constraint, as x_f is the shared variable to be optimized between the agents, and removing this constraint would lead each to solve their solution entirely independently. Constructing the Lagrangian with just the coupling constraint, one obtains the equation:

$$\begin{aligned} L(x, u, \lambda) &= \sum_{i=1}^{N} \sum_{t=0}^{T_f-1} [x_i(t)^T x_i(t) + u_i(t)^T u_i(t)] + x_i(T_f)^T x_i(T_f) + \sum_{i=1}^{N} \lambda_i^T (x_i(T_f) - x_f) \\ &= \sum_{i=1}^{N} (\sum_{t=0}^{T_f-1} [x_i(t)^T x_i(t) + u_i(t)^T u_i(t)] + x_i(T_f)^T x_i(T_f) + \lambda_i^T x_i(T_f)) - \sum_{i=1}^{N} \lambda_i^T (x_f) \end{aligned}$$

The dual problem is then:

$$d(\lambda) = \arg\min_{x_f, u} \sum_{i=1}^{N} (\sum_{t=0}^{T_f-1} [x_i(t)^T x_i(t) + u_i(t)^T u_i(t)] + x_i(T_f)^T x_i(T_f) + \lambda_i^T x_i(T_f)) - \sum_{i=1}^{N} \lambda_i^T x_f$$

Examining the final term, it is clear that unless $\sum_{i=1}^{N} \lambda_i = 0$, this problem will be unbounded and will force $x_f \to \infty$. To combat this, we can add the dual constraint:

$$\sum_{i=1}^{N} \lambda_i = 0$$

The dual problem can now be expressed as:

$$d(\lambda) = \arg\min_{u} \sum_{i=1}^{N} (\sum_{t=0}^{T_f-1} [x_i(t)^T x_i(t) + u_i(t)^T u_i(t)] + x_i(T_f)^T x_i(T_f) + \lambda_i^T x_i(T_f))$$

Now the full dual problem can be expressed as:

$$\max_{\lambda} d(\lambda)$$
$$\text{s.t. } \sum_{i=1}^{N} \lambda_i = 0$$

To decompose the problem by agent, each agent can be though of as maximizing some $d_i(\lambda_i)$, which is defined as:

$$d_i(\lambda_i) = \arg\min_{u} \sum_{t=0}^{T_f-1} [x_i(t)^T x_i(t) + u_i(t)^T u_i(t)] + x_i(T_f)^T x_i(T_f) + \lambda_i^T (x_i(T_f) - x_f) \tag{2}$$

Projected Subgradient Method

The first method of solving this problem is the projected subgradient method, which uses the following algorithm:

- **Step 0:** Initialize some guess for the global variable x_f and each agent's λ_i. In the code, the variables are initialized as $x_f^0 = [0,0]^T \quad \& \quad \lambda_i^0 = [0,0]^T \quad \forall i$.
- **Step 1:** For each agent i, obtain the trajectory x_i^k and the control actions u_i^k by solve the subproblem:

$$\arg\min_{x_i, u_i} \sum_{t=0}^{T_f-1} [x_i(t)^T x_i(t) + u_i(t)^T u_i(t)] + x_i(T_f)^T x_i(T_f) + \lambda_i^T (x_i(T_f) - x_f^k)$$
$$\text{subject to: } x_i(0) = x_{0,i}$$
$$x_i(t+1) = A_i x_i(t) + B_i u_i(t) \quad \forall t$$
$$|u_i(t)| \leq u_{max} \quad \forall\, t$$

  In optimizing this problem, one obtains the $x_i^k(T_f)$ for each agent, which is each agent's "guess" at what the opimal final position is for that iteration.
- **Step 2:** Iterate on the new global variable x_f by taking the averages of each agent's $x_i^k(T_f)$ using the equation:

$$x_f^{k+1} = \frac{1}{4} \sum_{i=1}^{4} x_i^k(T_f)$$

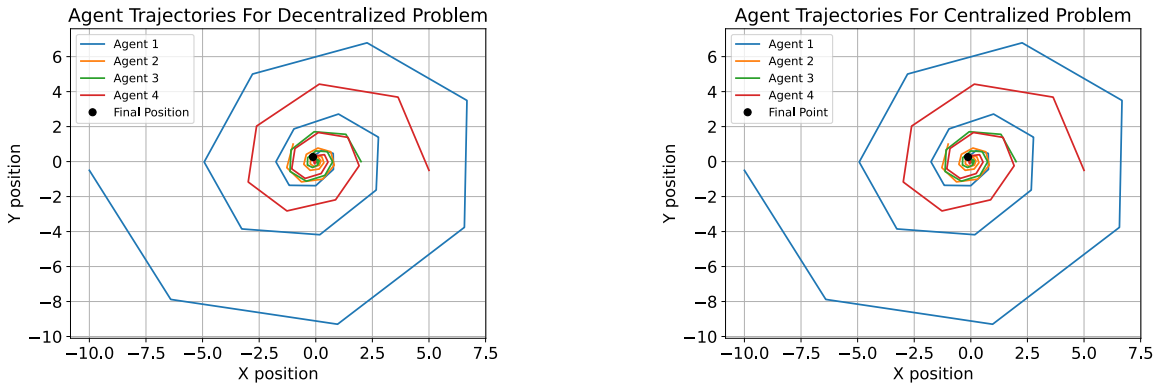
  One important note is that, in real life systems, this requires communication between all of the agents in the system or some centralized coordinator.
- **Step 3:** Use sub-gradient ascent for each agent's λ_i using the equation:

$$\lambda_i^{k+1} = \lambda_i^k + \alpha g_i$$

  where the gradient g_i is the gradient of $d(\lambda_i)$ with respect to λ_i, which is $g_i = (x_i^k(T_f) - x_f^k)$. The λ_i^{k+1} are then projected onto the possible set of λ_is defined by the constraint $\sum_{i=1}^{4} \lambda_i = 0$, using the equation:

$$\lambda_i^{k+1} = \lambda_i^{k+1} - \frac{1}{4}\sum_{i=1}^{4} \lambda_i^{k+1}$$

Following this algorithm should iteratively maximize the dual problem and therefore minimize the primal problem. The algorithm was implemented in python using `cvxpy` for optimizing Step 1. This was compared with the output that `cvxpy` had for optimizing the full primal problem, though a very large classical gradient descent algorithm could be used in the place of cvxpy for the centralized problem. The trajectories for both the Projected Subgradient and the cvxpy-solved centralized problem can be seen in Figure 1:



(a) Trajectory for Projected Subgradient Solution

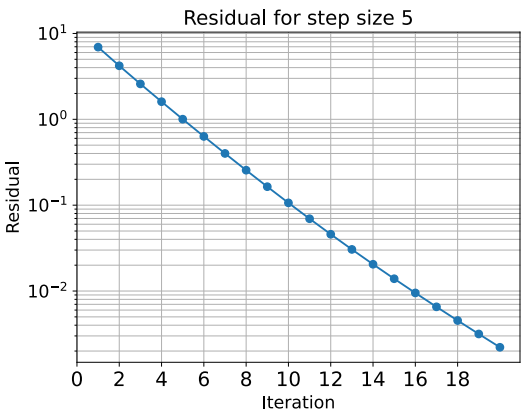
(b) Trajectory for Centralized Solution

Figure 1: Comparison of Trajectory Between the Projected Subgradient and Centralized Solutions to the Multi-Agent Rendezvous Problem

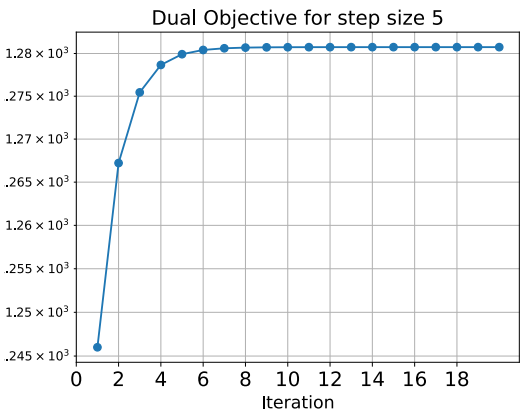
The subgradient method performs very similarly to the centralized optimum solution, with both methods resulting in almost the same exact point slightly positive in y and slightly negative in x. Both, as one would expect from the primal objective, result in endpoints quite close to the origin. The logarithmic error sequence can be calculated and normalized using the following equation:

$$\text{residual}_i^k = \frac{|x_i^k(T_f) - x_f^k|}{|x_f^k|} \tag{3}$$

for each agent i. In all optimization methods examined, the highest error for any agent was saved as the error for the system, called the residual. The result for the subgradient method where $\alpha = 5$ can be seen in Figure 2a alongside the evolution of the dual objective function:



(a) Logarithmic Error Sequence for The Subgradient Method where $\alpha = 5$.



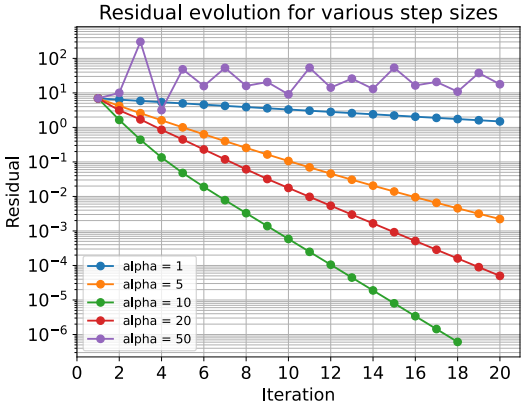
(b) Evolution of the Dual Objective for The Subgradient Method where $\alpha = 5$.

Figure 2: Performance of Projected Subgradient for $\alpha = 5$

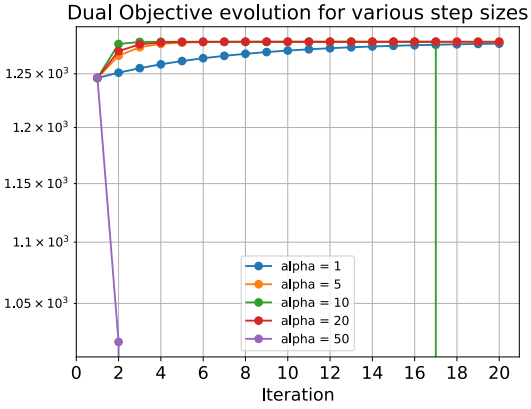
It is difficult to draw a conclusion from this before analyzing it in comparison to other methods, which will be examined in the following sections.

Step Size For Subgradient Method

To investigate the importance of the step size parameter in the dual decomposition with projected subgradient method, a convergence analysis and dual objective analysis was conducted and can be seen in Figures 3a and 3b:



(a) Comparison of Logarithmic Error Sequence for Multiples Values of α in the Projected Subgradient Method



(b) Comparison of Evolution of Dual Objectives for Multiple Values of α in the Projected Subgradient Method

Figure 3: Performance of Projected Subgradient for a Range of α Values

In all of the subsequent dual objective plots, at the iteration where a method converges to a specified tolerance, the dual convergence plot represents this with a vertical line in the color that represents the plot. The figure shows that the projected subgradient method is very sensitive to tuning of its parameter α. It can have high convergence rates at the right α values, evidenced by its convergence to $\frac{|x_i(T_f) - x_f|}{|x_f|} < 10^{-6}$in 18 steps at $\alpha = 10$, but if α is too low the convergence is extremely slow, and if α is too high the system can destabilize entirely. The $\alpha = 50$ case shows this destabilization happening. The almost vertical purple line representing $\alpha = 50$ is in fact not the vertical line representing convergence, but is instead the huge downward jump of the dual objective which it is attempting to maximize.

Incorporating Nesterov's Method for Acceleration

In order to accelerate the convergence of the dual function, the so-called Nesterov's Method can be applied to Step 3 of the projected subgradient method.

The general concept of Nesterov's Method is to introduce "momentum" into the gradient ascent loop via an intermediate λ_i^{inter}. It depends on a parameter L, which is supposed to be the Lipchitz constant assuming a strongly convex and smooth function. Another parameter m is supposed to be the modulus of the strongly convex function, though both the parameters L and m can be iterated through to find suitable values to accelerate convergence. Using L and m, one can define $\alpha = \frac{1}{L}$ and $\beta = \frac{\sqrt{\frac{L}{m}}-1}{\sqrt{\frac{L}{m}}+1}$. Then, the sequence for updating each λ_i is as follows:

1. $\lambda_i^{inter} = \lambda_i^k + \beta(\lambda_i^k - \lambda_i^{k-1})$
2. $\lambda_i^{k+1} = \lambda_i^{inter} + \alpha g_i^k$

Where g_i is the same gradient calculated for the projected subgradient method. It is important to note that generally, g_i would be replaced with $\nabla d(\lambda_i^{inter})$ in Nesterov's method, but the gradient is not a function of lambda in this particular case, so the same g_i is used. This two-step process replaces step 3 of the traditional projected subgradient method. The same residual function was used, and the residual and dual for the accelerated function can be seen alongside the traditional subgradient method in Figures 4 and 5:

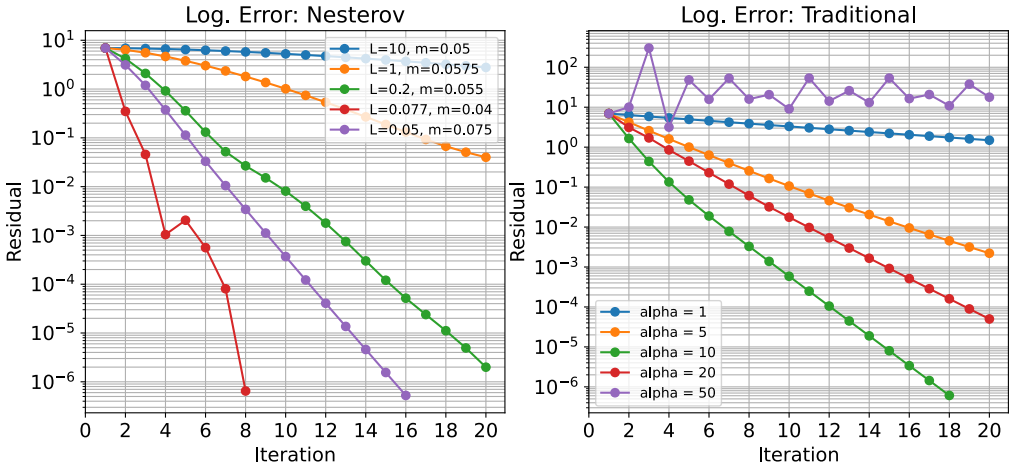


Figure 4: Comparison of Logarithmic Error Sequence for Nesterev-accelerated and original subgradient method over a range of parameters

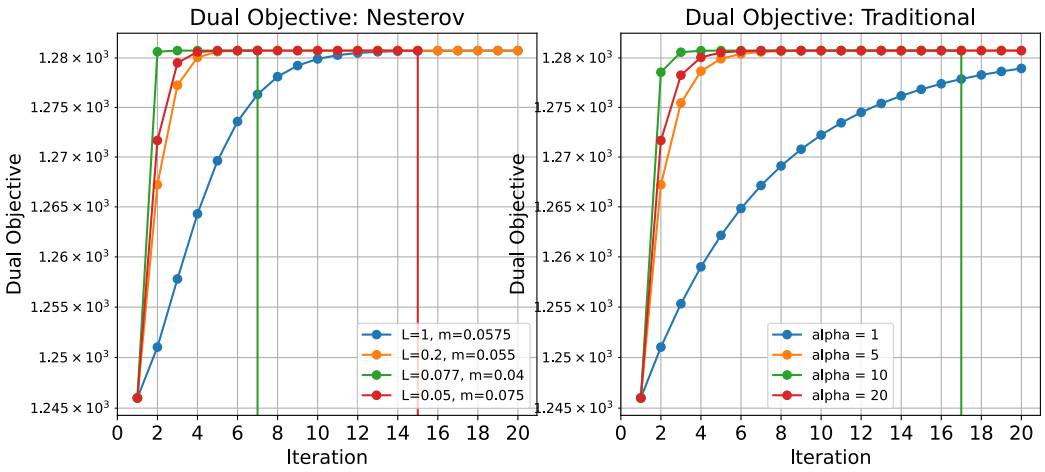


Figure 5: Comparison of Dual Objective Evolution for Nesterev-accelerated and original subgradient method over a range of parameters

Figure 4 shows that Nesterov's method displays significant possible decrease in the number of iterations to convergence, with the fastest convergence occurring in eight iterations compared to the eighteen iterations for the fastest non-accelerated method, a 225% increase in speed. From the dual objective it is somewhat harder to spot, but there is a markedly higher convergence rate onto the dual objective as well. One important caveat is that, like the non-accelerated method, it is very sensitive to its parameters. If the function is not m-strong convex and L-smooth, finding an optimal combination of L and m can be challenging, as small changes in both make large differences in convergence rate and stability. For example, at L=0.077 and m=0.04, the system converges in 8 iterations; quite fast. At L=0.05 and m=0.005, though, the system is entirely unstable as seen in Figure 6, where the residual has a saw-shaped increase and the trajectories do not converge at all in the end.

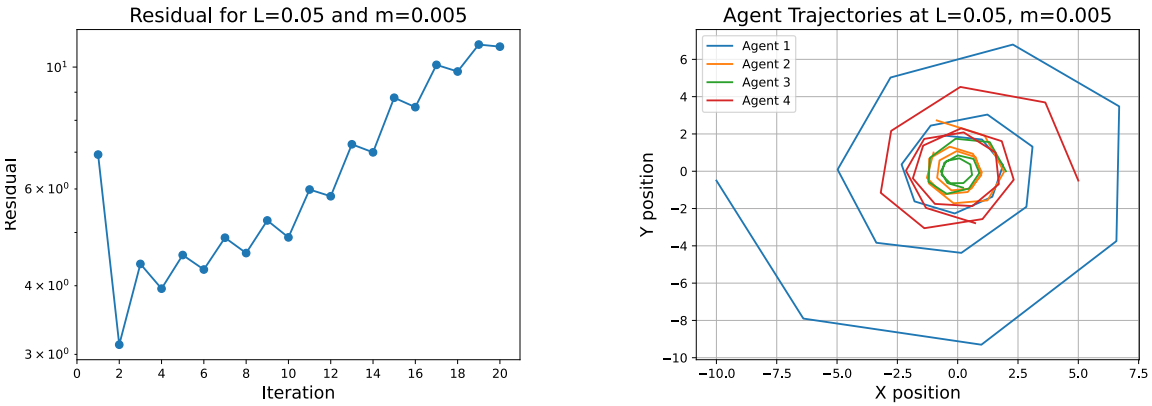


Figure 6: Residual and Dual Objective Evolution for L=0.05 and m=0.005. The system becomes unstable.

Though Nesterov's method is a powerful tool, it clearly requires careful analysis of the parameters L and m before implementation.

Implementing Consensus Approach

The final alteration on the subgradient approach is the consensus method. One big limitation of the traditional subgradient approach discussed in previous sections is that in Step 2, the method requires all agents to be connected to each other or some centralized agent in order to take the average of x_f. It is often beneficial, if not necessary, to have a more flexible communication algorithm in order to scale the system with more relaxed communication constraints and without the need for a centralized

agent. The consensus approach solves this problem by designating a consensus matrix W that defines a weighted sum of averaging between agents, and therefore describes the necessary communication structure between agents. The consensus matrix used is:

$$W = \begin{bmatrix} 0.75 & 0.25 & 0 & 0 \\ 0.25 & 0.5 & 0.25 & 0 \\ 0 & 0.25 & 0.5 & 0.25 \\ 0 & 0 & 0.25 & 0.75 \end{bmatrix}$$

From this W matrix, it can be seen that Agent 1 is connected to Agent 2, Agent 2 is connected to Agents 1 and 3, Agent 3 is connected to Agents 2 and 4, and Agent 4 is only connected to Agent 3. In the consensus approach, Step 2 of the subgradient method takes the form:

$$\begin{bmatrix} x_{f,1}^{k+1} \\ x_{f,2}^{k+1} \\ x_{f,3}^{k+1} \\ x_{f,4}^{k+1} \end{bmatrix} = W^{\gamma} \begin{bmatrix} x_1^k(T_f) \\ x_2^k(T_f) \\ x_3^k(T_f) \\ x_4^k(T_f) \end{bmatrix}$$

Where γ represents the number of consensus rounds that the system will carry out for each timestep. A consensus round can be though of as a single iteration where Agents share their current values with their neighbors and take their weighted sums. An important characteristic of a consensus matrix is that its rows and columns all add to 1. In addition, each agent has some pathway to each other agent. Because of this, the limit of the expression W^{γ} as $\gamma \to \infty$ goes to a matrix with all elements equal to $\frac{1}{n}$ where n is the number of rows/columns in the square matrix. This means that as $\gamma \to \infty$, the next steps for the shared parameter x_f approaches the average of all $x_i(T_f)$ as in the previous methods. Using this approach, the logarithmic error sequence can be seen for a range of γ values and $\alpha = 5$ in Figure 7:

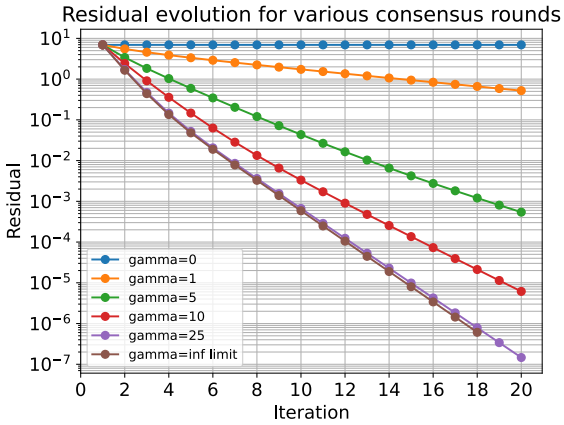


Figure 7: Residual Evolution for a Range of γ Values For $\alpha = 5$

As expected, the process performs better and better the more consensus rounds are used, converging eventually to the case where $x_f = \frac{1}{4}\sum x_i(T_f)$. After 25 consensus steps the performance is almost indistinguishable from the theoretical $\gamma \to \infty$ case in which x_f^k is calculated the by averaging the $x_i^k(T_f)$ values of each agent.

Question 2: Alternating Direction Method of Multipliers(ADMM)

The ADMM Method differs in a few key ways from the subgradient method in the first question. Recall the primal problem:

$$\begin{aligned}\min_{x_f,u_i} \sum_{i=1}^{N}\sum_{t=0}^{T_f-1}[x_i(t)^T x_i(t)+u_i(t)^T u_i(t)]+x_i(T_f)^T x_i(T_f)\\ \text{subject to:}\quad x_i(t+1)=A_i x_i(t)+B_i u_i(t)\quad \forall i,t\\ |u_i(t)|\le u_{max}\quad \forall\, i,t\\ A_i^{T_f}x_{0,i}+\sum_{k=0}^{T_f-1}A_i^{T_f-1-k}B_i u_i(k)-x_f=0\quad \forall i\end{aligned}$$

The original method was to construct the Lagrangian in Equation 2. ADMM uses an augmented Lagrangian that multiplies the square of the Euclidian norm of the coupling term by a parameter $\frac{\rho}{2}$, such that the augmented Lagrangian is:

$$\begin{aligned}L(x,u,\lambda)=\sum_{i=1}^{N}\sum_{t=0}^{T_f-1}[x_i(t)^T x_i(t)+u_i(t)^T u_i(t)+\lambda_i^T(x(T_f)-x_f)+\frac{\rho}{2}||x_i(T_f)-x_f||_2^2]+x_i(T_f)^T x_i(T_f)\\ =\sum_{i=1}^{N}\sum_{t=0}^{T_f-1}[x_i(t)^T x_i(t)+u_i(t)^T u_i(t)+\lambda_i^T(x(T_f)-x_f)+\frac{\rho}{2}(x_i(T_f)-x_f)^T(x_i(T_f)-x_f)]+x_i(T_f)^T x_i(T_f)\end{aligned}$$

Using this augmented Lagrangian, one can define the dual problem for each agent as:

$$d_i(\lambda)=\arg\min_{u_i}\sum_{t=0}^{T_f-1}[x_i(t)^T x_i(t)+u_i(t)^T u_i(t)+\lambda_i^T(x(T_f)-x_f^k)+\frac{\rho}{2}||x_i(T_f)-x_f^k||_2^2]+x_i(T_f)^T x_i(T_f)$$

Solving this augmented subproblem for each agent is the first step of each iteration for the ADMM approach, and is done using `cvxpy` in the python implementation. When each subproblem is solved, the $x_i^{k+1}(T_f)$ is obtained and used to calculate the next consensus variable using the equation:

$$x_f^{k+1}=\frac{1}{4}\sum_{i=1}^{4}(x_i^{k+1}(T_f)+\frac{1}{\rho}\lambda_i^k)$$

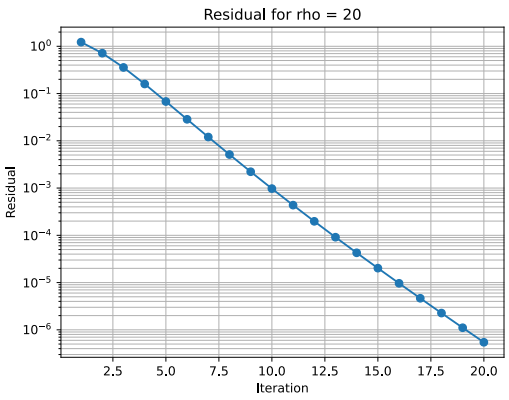
Then, using the new x_f^{k+1}, one can update the Lagrange multipliers using the equation:

$$\lambda_i^{k+1}=\lambda_i^k+\rho(x_i^{k+1}(T_f)-x_f^{k+1})$$

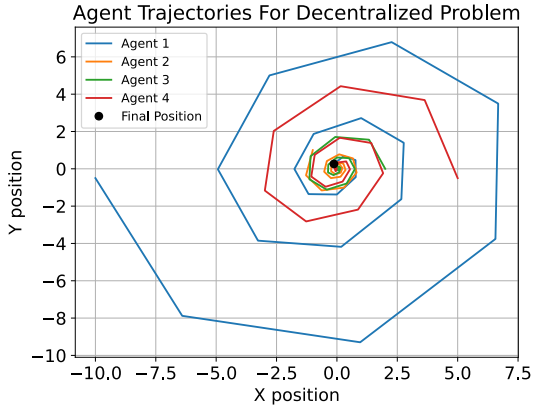
Since $(x_i^{k+1}(T_f)-x_f^{k+1})$ is the coupling term.

ADMM Convergence Analysis

This method was fully implemented in Python and simulated with the value $\rho=20$, and the resulting logarithmic error plot and trajectory can be seen in Figures 8a and 8b:



(a) Caption



(b) Caption

Figure 8: Performance of ADMM for $\rho = 5$

For $\rho = 20$, the ADMM method performs quite well, converging to $\frac{|x_i(T_f) - x_f|}{|x_f|} < 10^{-6}$ in 20 iterations, within 2 iterations of the non-accelerated subgradient method. Comparing the final trajectory to the centralized optimal trajectory in Figure 1b, the ADMM trajectory also performs very well, with an endpoint in almost the same location in x and y, though inspecting particular agent trajectories like Agent 4, one notes some slight differences in individual waypoints that evidence a sub-optimal solution. Because the dual cost function is different for the ADMM and the other methods examined, no dual objective comparison is made.

Effect of Changing ρ

As discussed for previous methods, parameters can often be very sensitive to small changes, even having the ability to destabilize the system entirely, so it is prudent to conduct an analysis on the effect ρ has on the convergence of the ADMM method. The ADMM method was tested over 6 ρ values and the logarithmic error sequence, calculated using Equation 3, can be seen in Figure 9

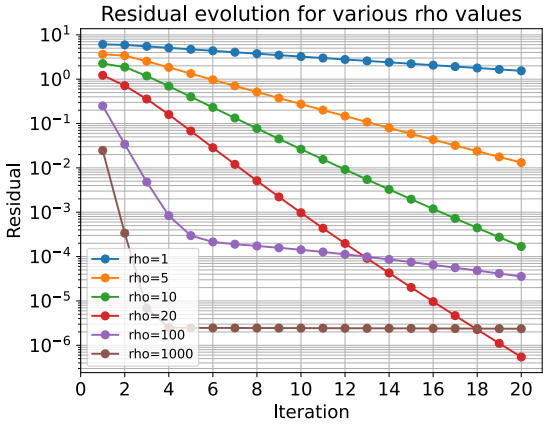


Figure 9: Caption

The figure shows one of the significant benefits of the ADMM method: stability. Its results are stable over a huge range of ρ values, with values from 1 to 1000 showing stability. As rho increases, the convergence speed increases, but on the other hand, as the convergence speed increases the convergence begins to plateau at slightly higher values. For example, the figure shows that at $\rho = 20$, it reaches $\frac{|x_i(T_f) - x_f|}{|x_f|} < 10^{-3}$ at the ninth iteration, while $\rho = 1000$ reaches $\frac{|x_i(T_f) - x_f|}{|x_f|} < 10^{-3}$ at the second iteration, or 450% faster. On the other hand, the $\rho = 1000$ case plateaus around $\frac{|x_i(T_f) - x_f|}{|x_f|} = 2 \times 10^{-6}$,

where the $\rho = 20$ case reaches below $\frac{|x_i(T_f)-x_f|}{|x_f|} = 10^{-6}$ and is still decreasing by the 20th iteration.

Comparing to other methods, the highest rates of convergence to errors around 10^{-3} is significantly better than all other methods. The highest convergence to 10^{-3} in any other method is the best performing Nesterov case, which converges to 10^{-3} in 6 iterations, or 3 times slower than the best performing ADMM case. At the same time, ADMM is much more resilient to changes in its parameter ρ, whereas the projected subgradient method is quite sensitive to α, and Nesterov's method is incredibly sensitive to L and m changes. Again, though, the Nesterov-Accelerated Projected Subgradient Method performs better if the desired performance is incredibly precise, as with proper tuning it provides the fastest convergence to high precision errors around $\frac{|x_i(T_f)-x_f|}{|x_f|} < 10^{-6}$.

Ultimately, all of the methods of solving the finite time-optimal rendezvous have their benefits and downsides, so the choice likely lies in the requirements of the system. The centralized primal method produces an optimal solution, but requires a centralized agent, so is clearly not a decentralized solution by any definition of the word. The subgradient method is powerful and can converge quickly with a high-precision, especially when using Nesterov's method for acceleration. Its downsides are that it requires robust communication between all agents and it is quite sensitive to parameter tuning, and can be destabilized relatively easily. If the system requires more relatex communication constraints between the agents, then the Consensus method can be useful for allowing agents to only communicate with neighbors, especially if multiple consensus rounds can be implemented. ADMM is useful in that it converges the fastest to mid-precision errors in the $\frac{|x_i(T_f)-x_f|}{|x_f|} < 10^{-3}$ range and is incredibly robust to tuning of its parameter ρ, but it is not the fastest at converging to extremely high precisions.

Bibliography

[1] Terelius, H., Topcu, U., & Murray, R. M. (2011). Decentralized Multi-Agent Optimization via Dual Decomposition. IFAC Proceedings Volumes, 44(1), 11245-11251.https://www.diva-portal.org/smash/get/diva2:453798/FULLTEXT01.pdf

[2] Chen, Y. (2023). *Lecture 9–10: Accelerated Gradient Descent. Lecture Notes.* https://pages.cs.wisc.edu/~yudongchen/cs726_sp23/Lecture_9_10_accelerated_GD.pdf