Delft University of Technology

Networked and distributed control systems
SC42101

Assignment 4

Authors:
Mihaly Fey (5476747)

June 20, 2025

1 Problem 1

The optimization problem can be observed in Equation 1. Since the problem specifies to only include u and x_f as the optimization variables, x_i has to be expressed as a function of u_i and $x_i(0)$. This can be seen in Equation 2. Matrix M_i is of dimension $\mathbb{R}^{2T_{final} \times T_{final}}$, while vector b_i is of length $2T_{final}$. This makes it possible to express the $x_i(T_{final}$ as a matrix multiplication between the last two rows of M_i and u_i plus the last two elements of b_i, as can be seen in Equation 3.

$$\begin{aligned} \underset{u}{\text{minimize}}\ & f(x,u) \\ & x_i(T_{final}) = x_f, \\ & |u_i(t)| < u_{max}, \\ f(x,u) = \sum_i \sum_t & x_i(t)^T x_i(t) + u_i(t)^T u_i(t) \\ & x_i(t+1) = A_i x_i(t) + B_i u_i(t) \\ & \text{for } t = 0..T_{final} - 1 \\ & \text{for } i = 1..4 \end{aligned} \tag{1}$$

$$\begin{aligned} x_i(t_n) &= A_i^n x_i(0) + \sum_{k=0}^{n-1} A_i^k B_i u_i(k), \\ \xi_i &= \begin{bmatrix} x_i(1) \\ x_i(2) \\ \vdots \end{bmatrix} = M_i u_i + b_i \end{aligned} \tag{2}$$

$$x_i(T_{final}) = M_{i,2T_{final}-1:2T_{final},1:T_{final}} + b_{i,2T_{final}-1:2T_{final}} = M_{i,final} u_i + b_{i,final} \tag{3}$$

To perform dual decomposition there have to be no common optimization variables, which means that x_f has to be removed, with for example primal decomposition. In the case of this problem, this extra variable can be eliminated by constraining the final states directly to each other. To obtain the dual problem, the coupling constraints and the Lagrangian of the primal have to be found. They can be seen in Equation 4 and Equation 6 respectively. λ is an 8 long vector, for each element of the four final states.

$$\mathbf{h}(u) = \begin{bmatrix} x_1(T_{final}) - x_2(T_{final}) \\ x_2(T_{final}) - x_3(T_{final}) \\ x_3(T_{final}) - x_4(T_{final}) \\ x_4(T_{final}) - x_1(T_{final}) \end{bmatrix} = \mathbf{0} \tag{4}$$

$$\mathbf{h}_1(u_1) = \begin{bmatrix} x_1(T_{final}) \\ 0 \\ 0 \\ -x_1(T_{final}) \end{bmatrix} \quad \mathbf{h}_2(u_2) = \begin{bmatrix} -x_2(T_{final}) \\ x_2(T_{final}) \\ 0 \\ 0 \end{bmatrix} \mathbf{h}_3(u_3) = \begin{bmatrix} 0 \\ -x_3(T_{final}) \\ x_3(T_{final}) \\ 0 \end{bmatrix} \quad \mathbf{h}_4(u_4) = \begin{bmatrix} 0 \\ 0 \\ -x_4(T_{final}) \\ x_4(T_{final}) \end{bmatrix} \tag{5}$$

$$\mathcal{L}(u, \lambda) = \sum_i f_i(x) + \lambda^T \mathbf{h}(u) \tag{6}$$

Based on the Lagrangian, the dual function can be set up:

$$d(\lambda) = inf_{u \in \mathcal{U}} \sum_i f_i(u) + \lambda^T \sum_i \mathbf{h}_i(u_i) \tag{7}$$

The decoupled problems are the solution of the individual agents, and can be seen in Equation 8.

$$\begin{aligned} \underset{u_i}{\text{minimize}}\ & f_i(u_i) + \lambda^{iT} \mathbf{h}_i(u_i) \\ & u_i \leq u_{max} \\ & -u_i \leq u_{max} \\ f_i = \xi^T \xi + u_i^T u_i &= u_i^T (M_i^T M_i + I) u_i^T + 2 b_i^T M_i u_i + b_i^T b_i \end{aligned} \tag{8}$$

This subproblem can be solved with `cvxpy`, with λ as a parameter. The problem asks for the master problem to be solved with the projected subgradient method. Since the dual function is always concave, a gradient ascent

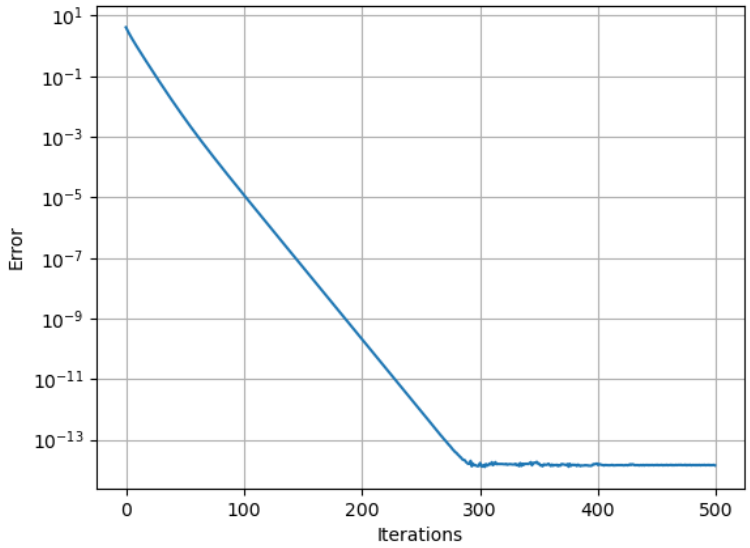

Figure 1: Convergence of the subgradient method, error measured from the centralized solution

can be performed. A subgradient of the dual function is given by solution of the sub-problems, $f_i(u_i^*)$ After all four subproblems have been solved, the dual variables are updated based on the following scheme:

$$\lambda_k + 1 = \lambda_k + \alpha(\sum_i h_i(u_i^*)) \tag{9}$$

Since there no λ values are constrained the projection step is not necessary and is not included in the update. The step size α is constant at first, set to 1. The method converges, as can be seen in Figure 1. In this graph and the following graphs the error is L_2 norm of the difference between the centralized and the converged distributed u input trajectories.

The trajectories of the converged solution can be seen in Figure 2. The converged solutions are very close to the centralised solution.

The final states also converge to the centralised values, as can be seen in

Investigating the effect of a varying step size the following square-summable series were used:

$$\alpha_k = 10/k \tag{10}$$

As the convergence slowed down at higher iterations numbers, the step size was multiplied by 10 to achieve similar convergence speeds as the constant step size. The error plots can be seen in Figure 4. Changing the step size for both fixed step results in faster convergence, until $\alpha \approx 6$, where the method stops converging and the error does not decrease anymore. With smaller step sizes, the error converges to $\approx 1e-14$, which is roughly around the machine precision, which explains why all step sizes stop converging at this error level. The variable step size iterations due to the high constant multiplier start off diverging, but quickly start converging and achieve very fast convergence rates, but as the step size starts diminishing convergence slows down for all step size multipliers.

Implementing an accelerated method was done with Nesterov's method. Applying it changes the λ update step to the following, with α being the the variable step size.

$$\begin{aligned} \gamma_k &= \lambda_k + \frac{\theta_k - 1}{\theta_{k+1}}(\lambda_k - \lambda_{k-1}) \\ \lambda_{k+1} &= \lambda_k + \alpha_k(\sum_i h_i(u_i^*(\gamma_k))) \\ \theta_0 = 0\,, \theta_{k+1} &= \frac{1 + \sqrt{1 + 4\theta_k^2}}{2} \end{aligned} \tag{11}$$

The error sequence plot can be seen in Figure 5. The accelerated method converges faster in the first few iterates. but the constant step size method converges to machine epsilon faster.

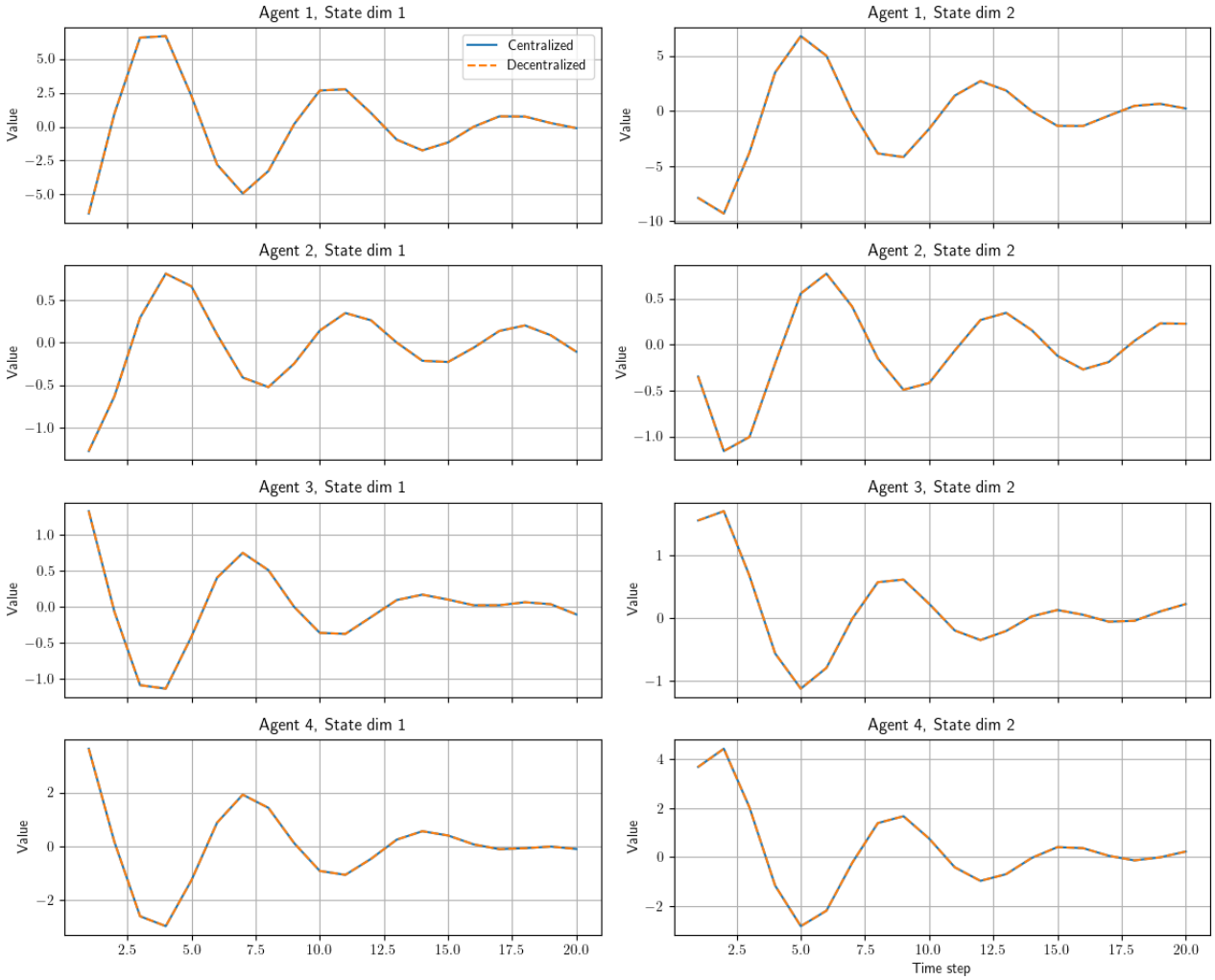

Figure 2: Agent state trajectories of the solution of the subgradient method

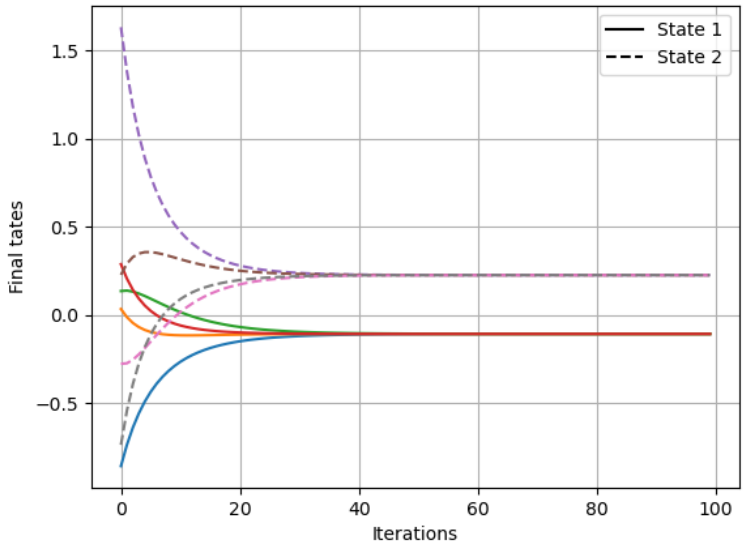

Figure 3: Convergence of the final states with the subgradient method

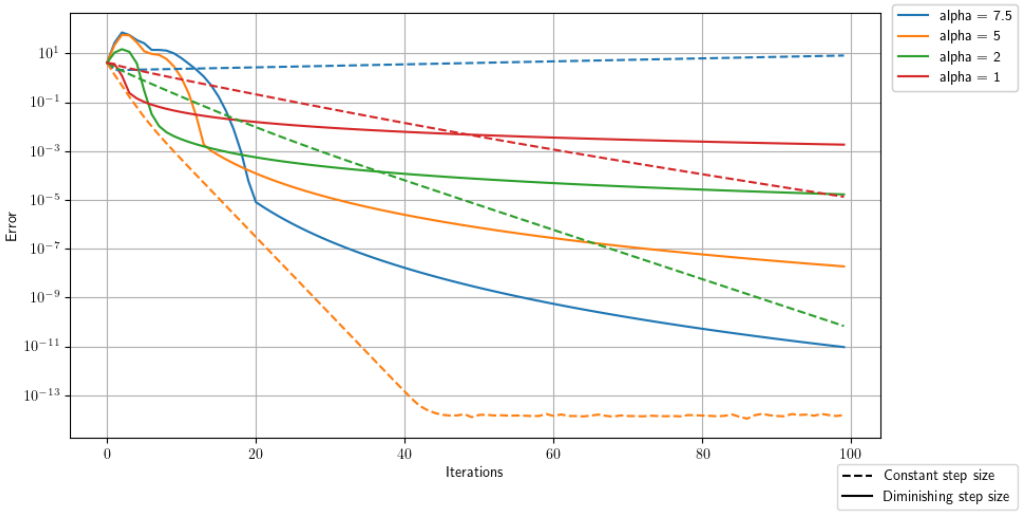

Figure 4: Effect of variable step size and step size scaling

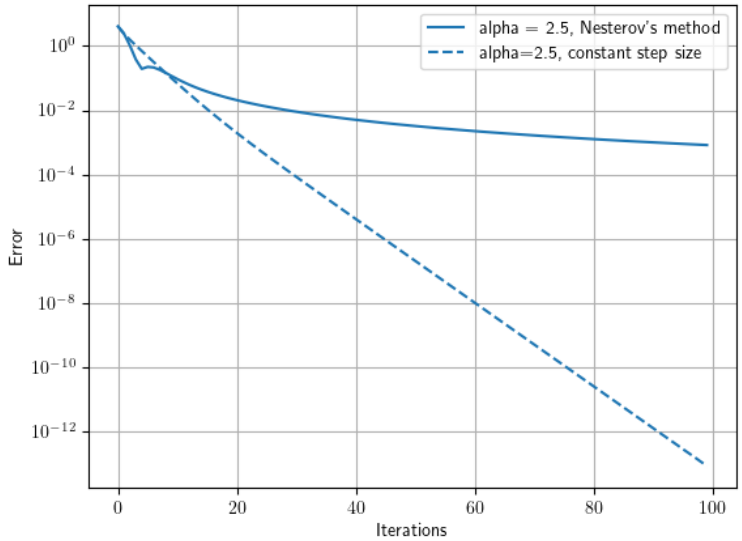

Figure 5: Nesterov's method showing increased convergence at the first few iterates

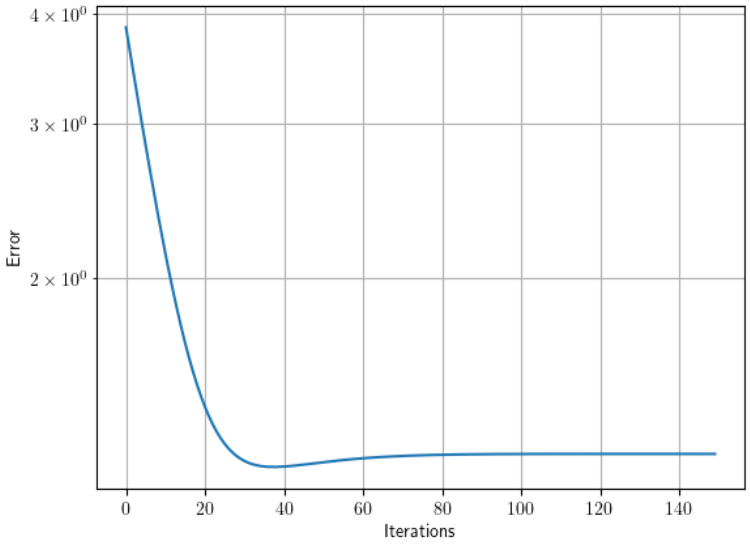

Figure 6: Convergence of ADMM

2 Problem 2

Implementing consensus ADMM involves adding back x_f optimzation variable, as $x_f = z$, the coupling variable. The structure with the subproblems stays, as the update of u_i happens simultaneously in each node. Then the update of the common variable, x_f, which appears as z in literature, happens, by averaging $x_i(T_{final})$ of all nodes. Then the dual variables are updated, to perform a cycle. The update three update steps are the following:

$$u_i^{k+1} = \underset{u_i}{\text{argmin}} \quad f_i(u_i) + \lambda^{kT} h_i^k(u_i) + \frac{\rho}{2}||h_i^k(u_i)||_2^2 \tag{12}$$

$$x_f^{k+1} = \frac{1}{4}\sum_i (x_i(T_{final}) + \frac{1}{\rho}\lambda_i^k)) \tag{13}$$

$$\lambda_i^{k+1} = \lambda_i^k + \rho h_i^{k+1} \tag{14}$$

, where

$$\lambda = \begin{bmatrix} \lambda_{1,1} & \lambda_{1,2} & \lambda_{2,1} & \lambda_{2,2} & \lambda_{3,1} & \lambda_{3,2} & \lambda_{4,1} & \lambda_{4,2} \end{bmatrix}^T \tag{15}$$

$$h_i^k = x_i^k(T_{final}) - x_f^k \tag{16}$$

$$x_i^k(T_{final}) = M_i u_i^k + b_i \tag{17}$$

The convergence plot of ADMM can be seen in Figure 6. The solution converges to a slightly different optimum than the centralized solution. The converged agent state trajectories can be seen in Figure 7. The final state constrain is held among the agents, so the consensus about the final state is established.

Changing the parameter affects the speed of the convergence, as can be seen in Figure 8. Putting the value higher that what is in the plot result in oscillation around the optima.

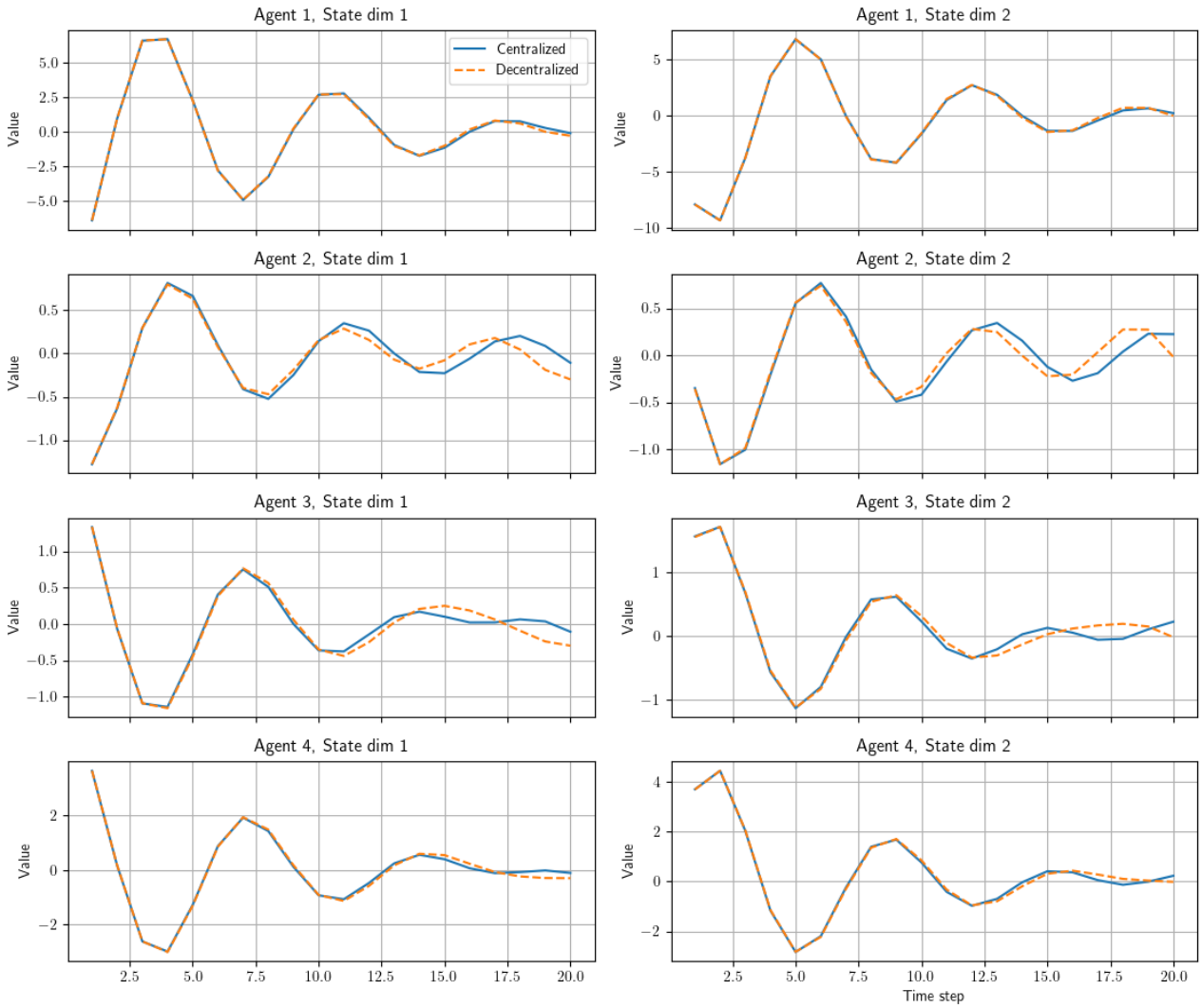

Figure 7: Agent State trajectories with ADMM

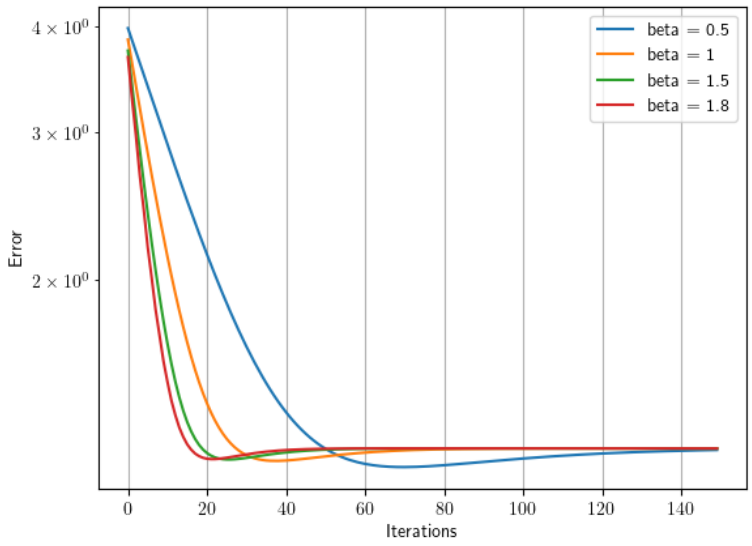

Figure 8: Effect of changing ρ on the convergence of ADMM