UNIVERSITÀ DI BOLOGNA

School of Engineering

Master Degree in Automation Engineering

Distributed Control Systems

Project 6: Distributed Dual Gradient Tracking for Microgrid Control

Professor: **Giuseppe Notarstefano**

Students: Baljinder Singh Bal
Francesco Taurone
Marco Toschi
Sean Brennan

Academic year 2018/2019

Abstract

The goal of this project is to solve two optimisation problems by means of distributed algorithms. In both tasks, multiple agents are present in a network and perform local computations while communicating with neighbouring agents to solve the overall problem. First, a *cost coupled* maximisation problem is solved by applying the algorithm presented in Qu and Na Li [2]. Then a *constraint coupled* minimisation problem is considered. In order to find a distributed optimisation algorithm, the structure of the functions and of the constraints is exploited and the associated dual problem is found. The algorithm implemented in the first part is further developed to solve this dual problem showing the expected convergence of all the chosen performance parameters. To verify the correctness of the algorithms, simulations are performed and the results presented using MATLAB.

Contents

List of Figures

Introduction

In order to solve a constrained coupled optimisation problem, well known centralised methods are available in literature. In general, constrained optimisation problem can possibly be solved with methods similar to the feasible direction descent method or with gradient projection techniques. Here we develop a solution based on duality theory that exploits the structure of the problem in order to derive a distributed solution. In this framework, graph theory is used to model the communication between agents and local computations are performed using information coming only from neighbouring agents. This allows the algorithm to solve the minimisation problem in a distributed fashion. In particular, in the first part of this project, a so called 'cost coupled' problem is solved by applying a well known result from the literature([2]. In the second part of this project a 'constrained coupled' optimisation problem arising in microgrid control is solved(). The distributed algorithm implemented in the first part is then adapted to suitably solve the dual maximisation problem associated with the primal minimisation problem. To show the correctness of the algorithm, a MATLAB implementation is used to perform simulations.
A MATLAB implementation of the algorithm presented in Qu and Na Li [2] is performed on the cost-coupled concave optimisation problem (Task 1). Then, a constrained-coupled optimisation problem presented in Notarnicola and Notarstefano [1] is analysed and the dual function is found (Task 2). Finally, the algorithm implemented in Task 1 is adapted to solve the dual problem (Task 3) and the simulation results are presented.

Motivations

Given the growing global demand for energy it is becoming more important to have intelligent energy systems such as the smart grid. In many real world applications and in industry keeping energy costs to a minimum is very important. To do so, it is essentially an optimisation problem. This gives motivation to this paper where the optimisation of a microgrid is considered with several cost constraints imposed.

Contributions

A method based on duality theory is used to solve a 'constrained coupled' optimisation problem. The dual function is formed by a sum of local functions and is maximised using only local computation and communication. The smoothness is exploited to obtain a fast convergence of the dual problem solution while also assuring primary recovery. The algorithm that is used is a combination of gradient descent and a gradient estimation scheme that utilises previously computed information to achieve fast and accurate estimation of the average gradient. The accuracy of this procedure is tested on the dual problem related to a minimisation problem involving coupling constraints.

Chapter 1

Task 1 - The Concave Problem

1.1 Introduction

In this first chapter the general problem is described and the algorithm to solve it is outlined. We consider a network of N agents each associated with a strictly convex function

$$q_i : \mathbb{R}^T \to \mathbb{R} \qquad \forall i \in \{1, ..., N\} \tag{1.1}$$

with local communication and computation. The communication is modelled by an *undirected* and *connected* graph $\mathbf{G} = (I, E)$ where I is the set of N nodes and $E \subset I \times I$. Agent i communicates with agent j if and only if $(i, j) \in E$. Moreover $w_{ij} = 0 \quad \forall (i, j) \notin E$.

1.2 Problem Setup

The optimisation problem that we want to solve is defined as

$$\max_{\lambda} \sum_{i=1}^{N} q_i(\lambda) \qquad\qquad i \in \{1, ..., N\} \tag{1.2}$$

Where $\lambda \in \mathbb{R}^T$ [1]. The cost function is the sum of quadratic functions

$$q_i(\lambda) = -(\lambda^T Q_i \lambda + r_i^T \lambda) \qquad\qquad i \in \{1, ..., N\} \tag{1.3}$$

and matrices Q_i are positive definite.
In order to solve this problem we exploit a method to solve a similar problem

[1] In the description of Task 1, we use T instead of S (used in the paper) to avoid confusion with the gradient tracking vector.

(1.4) presented in Qu and Na Li [2] which makes use of the distributed algorithm (1.5) and (1.6) with two updates per agent.

$$\min_x f(x) \triangleq \min_x \frac{1}{N} \sum_{i=1}^{N} f_i(x) \tag{1.4}$$

$$x_i^{t+1} = \sum_{j=1}^{N} w_{ij} x_j^t - \eta s_i^t \tag{1.5}$$

$$s_i^{t+1} = \sum_{j=1}^{N} w_{ij} s_j^t + \nabla f_i(p_i^{t+1}) - \nabla f_i(p_i^t) \tag{1.6}$$

Where x_i^t is the estimate at time t of the minimum of the global minimisation problem of agent i, N is the number of agents, w_{ij} is the ij^{th} entry of the consensus weighed matrix W, which is doubly stochastic, η is the fixed step size, s_i is the estimate of the average gradient $\frac{1}{N}\sum_{i=1}^{N} \nabla f_i(x_i(t))$. The algorithm is initialised by taking an arbitrary value of $x_i(0)$, and $s_i(0) = \nabla f_i(x_i(0))$. The original problem (1.2) is then adapted to (1.7) to allow the application of algorithm described by the updates (1.6) and (1.5).

$$\min_\lambda \frac{1}{N} \sum_{i=1}^{N} \bar{q}_i(\lambda) \tag{1.7}$$

Where $\bar{q}_i(\lambda) = -N q_i(\lambda)$. It follows that the updates that each agent must perform can be described as

$$\lambda_i^{t+1} = \sum_{j=1}^{N} w_{ij} \lambda_i^t - \eta s_i^t \tag{1.8}$$

$$s_i^{t+1} = \sum_{j=1}^{N} w_{ij} s_j^t + \nabla \bar{q}_i(\lambda_i^{t+1}) - \nabla \bar{q}_i(\lambda_i^t) \tag{1.9}$$

Where the gradient of $\bar{q}_i$ is

$$\nabla \bar{q}_i(\lambda) = -N \nabla q_i(\lambda) \tag{1.10}$$

and in particular, in our case we have

$$\nabla \bar{q}_i(\lambda) = N(2Q_i\lambda + r_i) \tag{1.11}$$

The initial values are set accordingly, taking any λ_i^0 and $s_i^0 = -N\nabla q_i(\lambda^0)$.

1.3 Implementation

The algorithm is implemented using MATLAB and all variables are randomly generated. In the following we report the MATLAB structures used for the implementation. We consider the dimension of each vector λ_i and s_i to be T. N is the number of agents in the communication graph and MAXITERS is the maximum number of iterations the algorithm runs for.

1.3.1 Q Matrix

Each agent has its own Q_i matrix, positive definite and diagonal with randomly generated positive entries. All the different Q_i matrices are stacked in a 3D matrix, Q, with each layer associated with a different agent. The dimension of Q is (T x T x N) with the third index indicating different agents.

1.3.2 R Matrix

Each agent has its own r_i vector. All the different r_i vectors are combined in a 2D matrix, R, with each column associated with a different agent. The dimension of R is (T x N) with the second index indicating different agents. Each entry being randomly generated.

1.3.3 LM Matrix

The matrix LM has dimension (T x MAXITERS x N) and is initialised to have all zero elements. Each t^{th} column of the (T x MAXITERS) matrix LM(:, t, j) stores the vector of dimension T associated with the j^{th} agent λ_j^t.

1.3.4 S Matrix

The matrix S has dimension (T x MAXITERS x N) and is initialised to have all zero elements. Each t^{th} column of S(:, t, j) stores the vector of dimension T associated with the j^{th} agent s_j^t.

1.3.5 W Matrix

The weighted consensus matrix is generated using the code and procedure presented in the lectures of the course. It is chosen to be strongly connected.

1.3.6 Step Size

According to Qu and Na Li [2] the algorithm works for a constant step size which is fixed at the beginning of each simulation.

1.3.7 The algorithm in MATLAB

Once all the matrices and variables have been created and initialised properly, the algorithm runs with three loops. The outer loop runs from $t \in \{1, \text{MAXITERS} - 1\}$, the middle loop runs from $i \in \{1, N\}$ and the inner loop runs from $j \in \{1, N\}$. The outer loop iterates through the second dimension (time) of each matrix while middle and inner loops go through agents and states.
To check the obtained results the optimum values are computed using also the centralised algorithm provided by MATLAB function "quadprog". The error between both results are plotted. These results are shown and discussed in Section 4.1.

Chapter 2

Task 2 -Dualisation of a Microgrid Control Problem

2.1 Developing the Algorithm

This task deals with the optimisation of a microgrid control with different costs and constraints. The problem considered is defined as

$$\min_{p_1,\dots p_N} \sum_{i=1}^{N} f_i(p_i) \tag{2.1}$$

$$\begin{aligned} \text{subj. to} \sum_{i\in\text{GEN}} p^{\tau}_{\text{gen},i} + \sum_{i\in\text{STOR}} p^{\tau}_{\text{stor},i} + \sum_{i\in\text{CONL}} p^{\tau}_{\text{conl},i} + p^{\tau}_{tr} - D^{\tau} = 0 \\ p_i \in X_i \\ \forall \tau \in [0, T] \\ \forall i \in \{1, ..., N\} \end{aligned}$$

The powers associated with each type of device in the microgrid has an associated cost and the task deals with the minimisation of the sum of such costs. The minimisation must also consider a coupling relationship (constraint) that relates the different powers of all the devices for each instant of time in a horizon from 0 to T. This can be written in a more compact form using a vector of dimension T for each agent by storing the powers at different times from 1 to T in a single vector. We can then rewrite the coupling equality constraint

$$\sum_{i\in\text{GEN}} p_{\text{gen},i} + \sum_{i\in\text{STOR}} p_{\text{stor},i} + \sum_{i\in\text{CONL}} p_{\text{conl},i} + p_{tr} - D = 0 \tag{2.2}$$

The goal now is to find the dual function of this constrained minimisation problem and to adapt so the distributed algorithm can be applied (1.8) and

(1.9) described previously. To do so, it is convenient to introduce another simplifying compact notation of the costs

$$h_i(p_i) = p_i, \qquad \forall\, i \in \text{GEN} \cup \text{STOR} \cup \text{CONL} \tag{2.3}$$

$$h_{tr}(p_{tr}) = p_{tr} - D \tag{2.4}$$

The problem (2.1) can now be restated

$$\min_{p_1,\dots p_N} \sum_{i=1}^{N} f_i(p_i) \tag{2.5}$$

$$\text{subj. to} \sum_{i=1}^{N} h_i(p_i) = 0$$

The Lagrangian function associated with this problem is

$$L((p_1,\dots,p_N),\lambda) = \sum_{i=1}^{N} f_i(p_i) + \lambda^T(\sum_{i=1}^{N} h_i(p_i)) \tag{2.6}$$

It follows that the dual function is

$$q(\lambda) = \inf_{p_i(j)\in X_i} \sum_{i=1}^{N}(f_i(p_i) + \lambda^T h_i(p_i)) = \sum_{i=1}^{N} q_i(\lambda) \tag{2.7}$$

$\forall i \in \{1,...,N\}$ and $\forall j \in \{0,...,T\}$. The dual problem can now be stated

$$\max_{\lambda}\ q(\lambda) \tag{2.8}$$

As the dual problem involves the maximisation of the dual function with respect to λ, we can rewrite it as the minimisation over λ of the negative of the dual function

$$\min_{\lambda}\ -q(\lambda) \tag{2.9}$$

With reference to the algorithm in Task 1, the problem can be restated in a form suitable for the direct application of the algorithm from Task 1

$$\min_{\lambda} \frac{1}{N} \sum_{i=1}^{N} \bar{q}_i(\lambda) \tag{2.10}$$

Where $\bar{q}_i(\lambda) = -Nq_i(\lambda)$

The update rules for each agent are defined by the algorithm

$$p_i^{t+1} = \arg \min_{p_i(j)\in X_i} (f_i(p_i) + (\lambda_i^t)^T h_i(p_i)) \tag{2.11}$$

$$\lambda_i^{t+1} = \sum_{i=1}^{N} w_{ij}\lambda_j^t - \eta s_i(t) \tag{2.12}$$

$$s_i^{t+1} = \sum_{i=1}^{N} w_{ij}s_j^t + \nabla\bar{q}_i(\lambda_i^{t+1}) - \nabla\bar{q}_i(\lambda_i^t) \tag{2.13}$$

$\forall i \in \{1, ..., N\}$ and $\forall j \in \{0, ..., T\}$. Where the initial conditions are fixed by picking any λ_i^0 and $s_i^0 = \nabla\bar{q}_i(\lambda_i^0)$. In particular, with reference to (2.3), the expression of the gradient for each agent is

$$\nabla\bar{q}_i(\lambda_i^t) = -Nh_i(p_i^t) = -Np_i^t \tag{2.14}$$

for all nodes except for the trade node which has the following expression for the gradient

$$\nabla\bar{q}_i(\lambda_i^t) = -N(h_i(p_i^t) - D) = -N(p_i^t - D) \tag{2.15}$$

Section 3 details the implementation of this algorithm in MATLAB and Section 4.2 presents the obtained results.

Chapter 3

Task 3 - Solving the Microgrid Control Problem

3.1 Problem set up

The algorithm described in Task 1 is adapted to solve problem (2.1). The first step is to model and generate the constraints for each cost, using a vectorial form.

3.1.1 Generator Cost

The generator has a cost function described in (3.1)

$$\sum_{\tau=0}^{T}[\alpha_1 p_{gen,i}^{\tau} + \alpha_2 (p_{gen,i}^{\tau})^2] \tag{3.1}$$

with constraints (3.2) and (3.3)

$$\underline{p} \leq p_{gen,i}^{\tau} \leq \bar{p}, \quad \forall \tau \in [0, T] \tag{3.2}$$

$$\underline{r} \leq p_{gen,i}^{\tau+1} - p_{gen,i}^{\tau} \leq \bar{r}, \quad \forall \tau \in [0, T-1] \tag{3.3}$$

The first constraint (3.2) is modelled by generating upper and lower bounds $\bar{P}$ and $\underline{P}$ respectively. Each in a column vector of dimension (T x 1) with randomly generated entries. Care is taken to ensure that the lower bound $\underline{P}$ is less that the upper bound $\bar{P}$ by making it negative while the upper bound $\bar{P}$ is positive.
The bounds $\bar{R}$ and $\underline{R}$ for the second constraint (3.3) are generated in the same way to those of (3.2) described above. To handle the $p_{gen,i}^{\tau+1} - p_{gen,i}^{\tau}$ part of the constraint it uses a shifted diagonal matrix B (3.5), where B has dimension (T-1 x T). The constraint (3.3) is therefore modelled as

$$\underline{R} \leq B p_{gen,i} \leq \bar{R} \tag{3.4}$$

$$B = \begin{pmatrix} -1 & 1 & 0 & \dots & 0 \\ 0 & -1 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \dots & -1 & 1 \\ 0 & \dots & \dots & 0 & -1 \end{pmatrix} \tag{3.5}$$

The final expression for generator's costs (3.1) is modelled in a vectorial form as

$$f_{gen,i} = \alpha_1 \mathbf{1}^T p_{gen,i} + \alpha_2 \, p_{gen,i}^T \, p_{gen,i} \tag{3.6}$$

Where **1** is a column vector of ones of dimension T.

3.1.2 Storage Cost

The storage agents have a cost function (3.7)

$$\varepsilon \, \|p_{stor,i}\|^2 \tag{3.7}$$

with constraints

$$-d_{stor} \leq p_{stor,i}^{\tau} \leq c_{stor}, \quad \forall \tau \in [0, T] \tag{3.8}$$

$$q_{stor,i}^{\tau+1} = q_{stor,i}^{\tau} + p_{stor,i}^{\tau}, \quad \forall \tau \in [0, T-1] \tag{3.9}$$

$$0 \leq q_{stor,i}^{\tau} \leq q_{max}, \quad \forall \tau \in [0, T] \tag{3.10}$$

The first storage constraint (3.8) is modelled in the same way as the first generator constraint (3.2) described in Section 3.1.1
The second one (3.9) is intended as a constraint on the sum of $p_{stor,i}^{\tau}$ for each time instant in the period. $q_{stor,i}^{\tau}$ represents the partial sum of those at time τ, with $q_{stor,i}^{0} = 0$. In order to model the sum in a vectorial fashion, a lower triangular matrix of ones and with dimension (T-1 x T) named B_{tri} is introduced. The sum is then modelled as

$$Q_{stor,i} = B_{tri} \, p_{stor,i} \tag{3.11}$$

where $Q_{stor,i}$ and $p_{stor,i}$ are the vectorial forms of $q_{stor,i}^{\tau}$ and $p_{stor,i}^{\tau}$ in a period.
The third storage constraint (3.10) is modelled in the same way as the first generator constraint (3.2) described in Section 3.1.1 and q_{max} is randomly generated and stacked in a vector of dimension (T-1, 1).
The final expression for storage costs (3.7) are modelled in a vectorial form as

$$f_{stor,i} = \varepsilon \, p_{stor,i}^T \, p_{stor,i} \tag{3.12}$$

3.1.3 Controllable Load Cost

The controllable load has a cost function

$$\sum_{\tau=0}^{T} \beta \max\{0, p_{des,i}^{\tau} - p_{conl,i}^{\tau}\} + \varepsilon \|p_{conl,i}\|^2 \tag{3.13}$$

with constraints [1]

$$-P_{conl}^{\tau} \leq p_{conl,i}^{\tau} \leq P_{conl}^{\tau}, \quad \forall \tau \in [0, T] \tag{3.14}$$

The constraint (3.14) is modelled in the same way as the first generator constraint (3.2) described in Section 3.1.1 with P being a vector with randomly generated entries.
The cost function (3.13) is modelled as

$$f_{conl,i} = \beta \max\{\mathbf{0}, P_{des,i} - p_{conl,i}\}^T \mathbf{1} + \varepsilon \, p_{conl,i}^T \, p_{conl,i} \tag{3.15}$$

where β and P_{des} are randomly generated.

3.1.4 Trade Cost

The trade node has a cost function

$$\sum_{\tau=0}^{T} (-c_1 p_{tr}^{\tau} + c_2 |p_{tr}^{\tau}|) + \varepsilon \|p_{tr,i}\|^2 \tag{3.16}$$

with constraints

$$-E \leq p_{tr}^{\tau} \leq E, \quad \forall \tau \in [0, T] \tag{3.17}$$

The constraint (3.17) is modelled in the same way as the first generator constraint (3.2) described in Section 3.1.1 with E being a vector of randomly generated entries.
The trade cost function (3.16) is modelled as

$$f_{trade,i} = -c_1 p_{tr} + c_2 |p_{tr}| + \varepsilon \, p_{tr,i}^T \, p_{tr,i} \tag{3.18}$$

3.2 Implementation

The problem is solved in two ways in MATLAB, using a centralised and a distributed method. The centralised method makes use of the CVX toolbox while the distributed uses the FMINCON function.

[1] P_{conl} is a vector of dimension (T+1, 1) with equal entries P_{conl}^{τ}. P_{conl}^{τ} refers to the P in the task assignment, in order to avoid confusion with the matrix P storing all the p_i^{τ} the notation is changed.

The number of generators, storage devices and controllable loads are chosen randomly, while there is always only one trade node. N is the sum of the number of devices, where

$$N = N_{gen} + N_{stor} + N_{conl} + N_{trade} \tag{3.19}$$

The LM, S and $grad_f$ matrices are set up and initialised in the same way as described in Section 1.2. Also matrices P, P_{error} and f are introduced. Matrices P and P_{error} have dimension (T x MAXITERS x N), f has dimension (1 x MAXITERS) and all are initialised to have all zero entries. The matrix P is used to record the minimisation of each node at each iteration. P_{error} is used to store the error at each iteration between the computed value of P and the optimal value P^* obtained in the centralised method.
The f row vector is used to store the full cost value at each iteration in the distributed method.

3.2.1 Centralised Method

The centralised method uses the CVX toolbox to solve the problem (2.1). The constraints are modelled and implemented as described in Section 3.1. CVX returns the Lagrange multipliers and the optimal P, P^* with the given constraints. The returned values are stored and compared with the values obtained from the distributed method.

3.2.2 Distributed Method

The distributed method begins by initialising the first columns of the S and P matrices. P is initialised by using an instance of FMINCON. S is initialised using [2]

$$S_i^1 = -NP_i^1 \tag{3.20}$$

and the initialised value of P, where i is the layer number in the 3D matrix. Differently, the initial value of the trade node is initialised using

$$S_i^1 = -N(P_i^1 - D) \tag{3.21}$$

Once the matrices have been initialised, the distributed method implements the algorithm used for Task 1 described in Section 1.2. Where $\nabla \bar{q}_i(\lambda_i^t)$ is now being computed using (2.14) for the generator, storage and controllable load nodes and using (2.15) for the trade node. There is an update to the P matrix at each time instant using the FMINCON function and the previously

[2] With S_i^1 meaning the column of agent i at the first iteration: S(:, 1, i) in MATLAB. The same notation applies to P_i^1 referring to $P(:, 1, i)$

calculated values of LM to calculate the gradient tracking.
f is updated at each iteration scuh that

$$f^t = \sum_{i=1}^{N} (f_{type,i}(P_i^t)) \tag{3.22}$$

where t is the time instant, i is the layer of the 3D matrix associated to the type of agent and subscript type is the node type. The costs need to be calculated differently depending on the type of node. Therefore P_{error} is updated at each iteration with the update rule

$$P_{error_i}^t = P_{opt_i} - P_i^t \tag{3.23}$$

The algorithm runs until the number of iterations (MAXITERS) is reached or until the maximum error between the norm of λ and λ^* is more that a defined threshold, where λ is calculated at each iteration in the distributed method and λ^* is the optimal vector computed with the centralised method. Once the simulation is stopped, the results between the distributed and centralised methods are compared and plotted in Section 4.2.

Chapter 4

Results

4.1 Task 1

4.1.1 Simulation Parameters

In the following we report the main parameters used for the simulations.
The step size is η=0.001 [1].
The number of nodes is N=5.
The dimension of the state of each agent is T=3.

4.1.2 Plots

The communication graph for Task 1 is shown in Figure 4.1. Each node also has a self edge that is not shown on the graph for lucidity. The weight of the self edge $w_{ii} = 1 - \sum_{\substack{j=1 \\ j \neq i}}^{N} w_{ij}$. In the following we consider as a performance metric the error on the Lagrange multiplier computed using a centralised algorithm and the estimate computed by each agent. In particular, in order to prove the convergence to an agreed vector of Lagrangian multipliers, the plot

$$\max_i ||\lambda_i^t - \lambda_{mean}^t||$$

displayed in Figure 4.2. This plot proves convergence by plotting the maximum norm of the error on lambda among all the agents. Therefore, if this function converges to zero, it means that all the other agents are converging too.

For proving optimality of the distributed solution, the

$$\max_i ||\lambda_i^t - \lambda^*||$$

[1] The step size has been empirically chosen by decreasing it until the convergence behaviour was satisfactory.

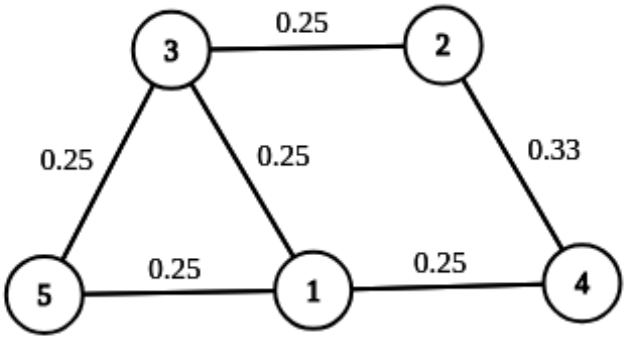

Figure 4.1: Graph of the agents of the experiment generated for task 1.

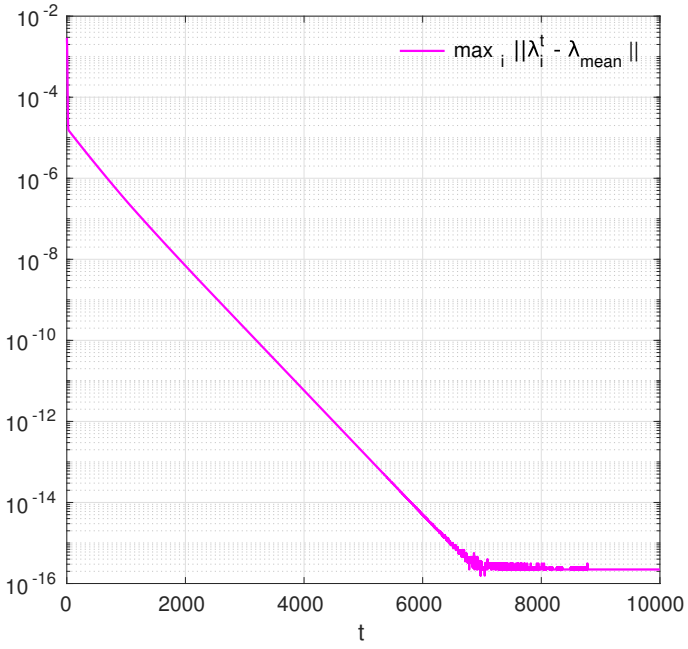

Figure 4.2: Task 1 - Evolution of $\max_i ||\lambda_i^t - \lambda_{mean}^t||$.

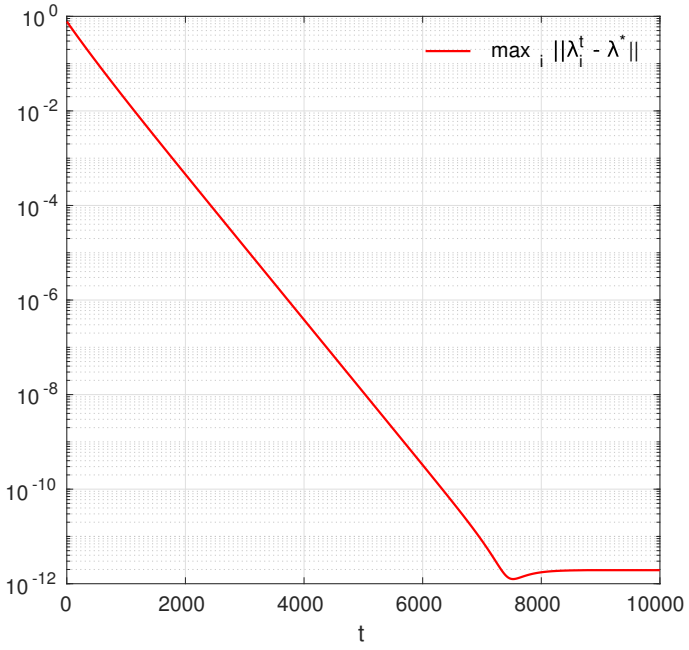

Figure 4.3: Task 1 - Evolution of $\max_i ||\lambda_i^t - \lambda^*||$.

should converge to zero, reported in Figure 4.3. As for the previous function, the graph plotting the error with respect to centralised Lagrangian multiplier proves that all the agents converge to the Lagrangian multiplier that makes primal recovery possible. From this convergence, it can be deduce that the optimisation problem in both centralised and decentralised version converge to the same solution of the minimisation.
It should be noted that since we proved the agreement of the agents of the Lagrangian multiplier, for the plot about the error with respect to the λ^* it would have been enough to prove the convergence of one of the agents.

4.2 Task 3

4.2.1 Simulation Parameters

In the following we report the main parameters used for the simulations.
The step size is η=0.001.
The number of generator nodes is N=4.
The number of storage nodes is N=3.
The number of controllable loads nodes is N=2.
The number of trade nodes is N=1.
The period is T=11. It has to be specified that the dimension of the state

is 12 since it is $T + 1$.

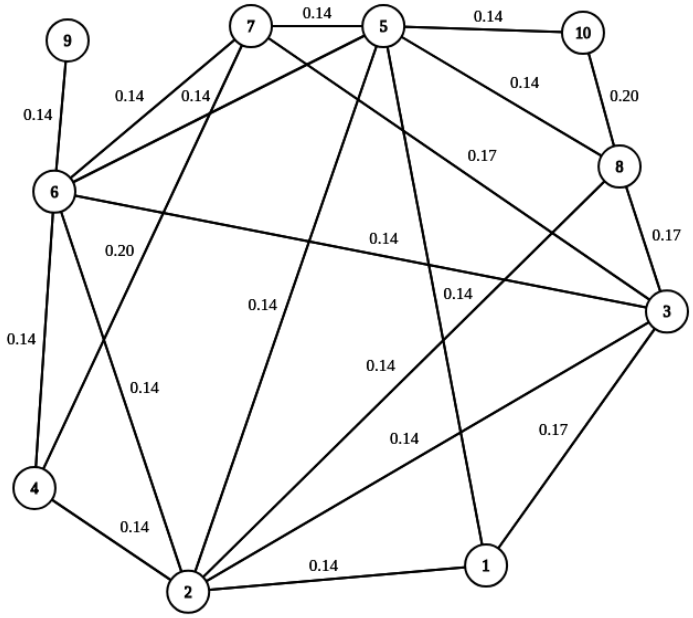

Figure 4.4: Graph of the agents of the experiment generated for Task 3.

Generator nodes corresponds to the nodes 1,2,3,4; Storage nodes corresponds to the nodes 5,6,7; Controllable load nodes corresponds to the nodes 9; Trade node corresponds to the node 10.

Figure 4.4 shows the digraph of the agents in Task 3 under the specified parameters. The graph is undirected, each edge has an associated weight. The sum of the weights on each node is equal to 1 as the adjacency matrix is doubly stochastic. Each node also has a self edge that is not shown on the graph for lucidity. The weight of the self edge $w_{ii} = 1 - \sum_{\substack{j=1 \\ j \neq i}}^{N} w_{ij}$.

4.2.2 Plots

Here again, in order to prove the convergence to an agreed optimal vector of Lagrangian multipliers, the plots $\max_i ||\lambda_i^t - \lambda_{mean}||$ and $\max_i ||\lambda_i^t - \lambda^*||$ are reported.
Additionally, the convergence to zero of the maximum norm of the vector S, approximating the average of the gradient of the dual function, shows

that indeed all the agents converge to an agreed point of minimum for the problem, since S drives the step size at each iteration shown in Figure 4.7. The value of the vector P which solves the distributed problem therefore converges to the centralised solution P^*, shown in Figure 4.8.
As a final check of the minimisation problem and of the associated primal recovery, Figure 4.9 shows the convergence of the minimum of the distributed solution of the cost with respect to centralised one. This once again proves that we reached the optimal solution.

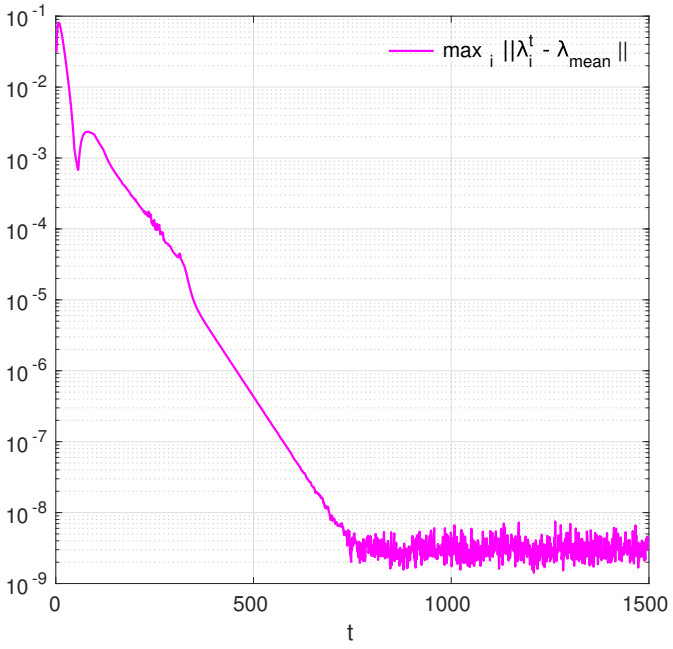

Figure 4.5: Task 3 - Evolution of $\max_i ||\lambda_i^t - \lambda_{mean}^t||$.

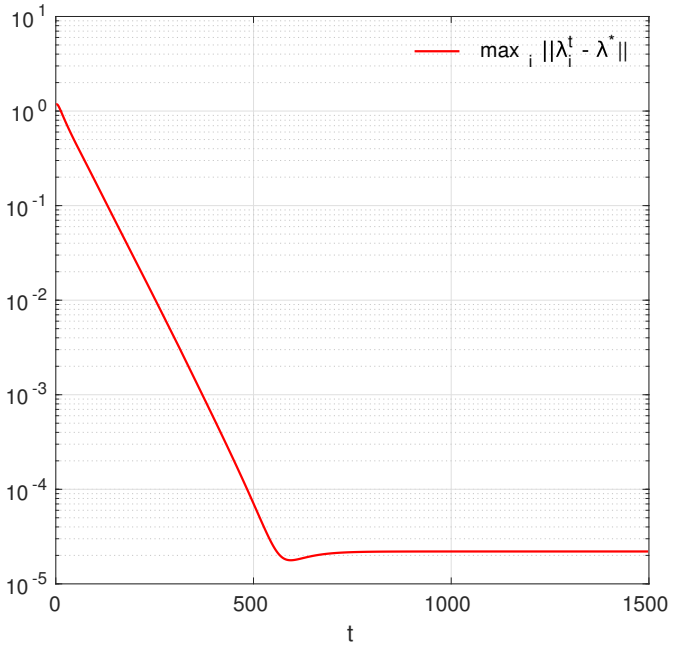

Figure 4.6: Task 3 - Evolution of $\max_i ||\lambda_i^t - \lambda^*||$.

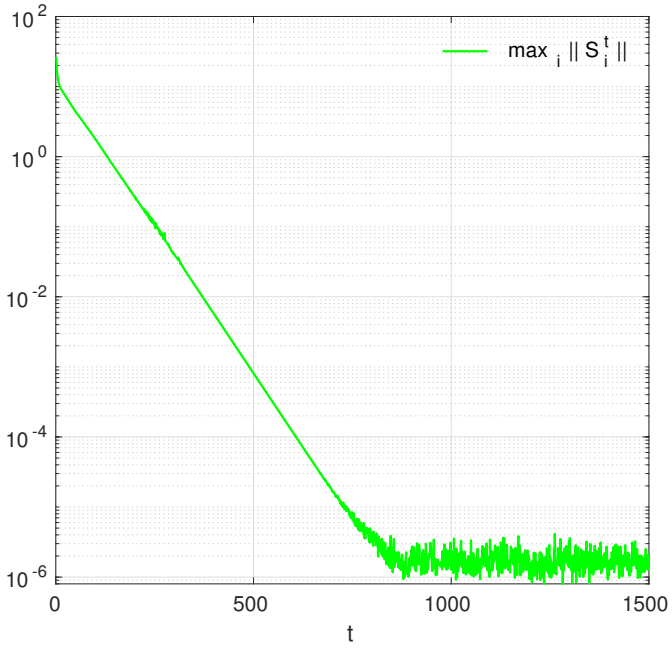

Figure 4.7: Task 3 - Evolution of $\max_i ||s_i^t||$.

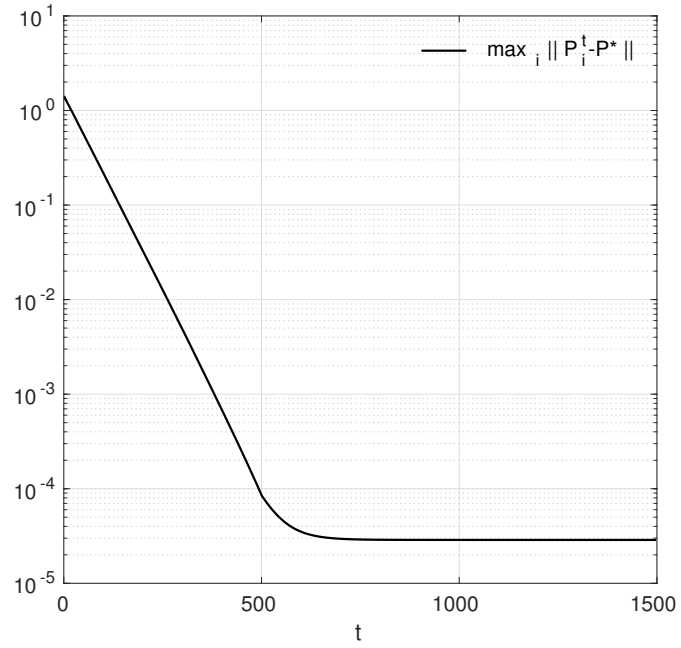

Figure 4.8: Task3 - Evolution of $\max_i ||p_i^t - p^*||$.

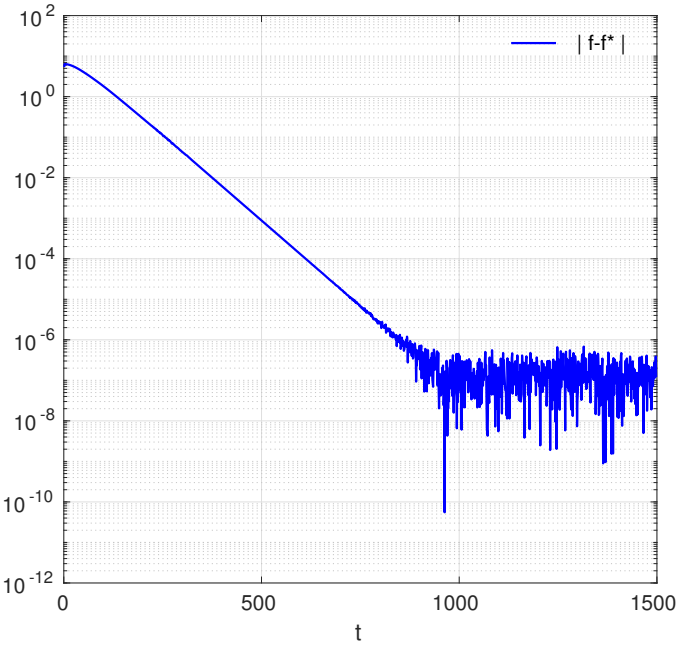

Figure 4.9: Task 3 - Evolution of $|f - f^*|$.

Chapter 5

Conclusions

A maximisation problem has been solved using the algorithm presented in Qu and Na Li [2]. Then, a minimisation problem based on the sum of local functions has been considered with the goal of solving it using only local computations and communications. In order to find a distributed optimisation algorithm the dual problem has been found and solved using the algorithm presented in Qu and Na Li [2]. The simulation has been carried out on MATLAB and the results presented, showing the convergence of the appropriate performance parameters.

Bibliography

[1] Ivano Notarnicola and Giuseppe Notarstefano. Constraint coupled distributed optimization: Relaxation and duality approach. *arXiv preprint arXiv:1711.09221*, 2017.

[2] Guannan Qu and Na Li. Harnessing smoothness to accelerate distributed optimization. *IEEE Transactions on Control of Network Systems*, 5(3):1245–1260, 2018.