Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 October 2021 doi:10.20944/preprints202110.0457.v1

Article

Generic framework for the optimal implementation of flexibility mechanisms in large-scale smart grids

Alejandro J. del Real [1], Andrés Pastor [2] and Jaime Durán [3,*]

[1] Department of Systems and Automation, University of Seville, Seville 41092, Spain
[2] IDENER, Seville 41300, Spain; andres.pastor@idener.es
[3] IDENER, Seville 41300, Spain; jaime.duran@idener.es
* Correspondence: adelreal@us.es

Abstract: This paper aims to provide the smart grid research community with an open and accessible general mathematical framework to develop and implement optimal flexibility mechanisms in large-scale network applications. The motivation of this paper is twofold. On the one hand, flexibility mechanisms are currently a hot topic of research, which is aimed to mitigate variation and uncertainty of electricity demand and supply in decentralised grids with a high aggregated share of renewables. On the other hand, a large part of such related research is performed by heuristic methods, which are generally inefficient (such methods do not guarantee optimality) and difficult to extrapolate for different use cases. Alternatively, this paper presents an MPC-based (Model Predictive Control) framework explicitly including a generic flexibility mechanism which is easy to particularise to specific strategies such as Demand Response, Flexible Production and Energy Efficiency Services. The proposed framework is benchmarked with other non-optimal control configurations to better show the advantages it provides. The work of this paper is completed by the implementation of a generic use case which aims to further clarify the use of the framework and thus, to ease its adoption by other researchers in their specific flexibility mechanisms applications.

Keywords: large-scale systems; aggregated constraints; aggregated terms; flexibility mechanisms; control algorithm; Model Predictive Control; Centralised MPC; Decentralised MPC; state-space model.

1. Introduction

Automating large-scale power and energy systems (flow control in chemical plants [1], autogenous grinding in large mines [2], smart electricity grids [3], etc.) is a process of great importance for today's industry. The need to control large networks in an optimal way in order to improve the efficiency, save costs and support the environment has become more widespread as systems have grown [4].

Historically, researchers and companies tried to design the best modelling and control techniques for these large grids and one of the most applied methods was Model Predictive Controls (MPC) [5],[6],[7]. This kind of controller tries to reach an optimal solution considering the current state of the system and the prediction of its near future behaviour [8]. It considers system and operational dynamics, predictions and constraints to determine optimal actions for present and future instants, that is, it computes the optimal control action based on the current state of the system, creating a prediction of optimal behaviour of its variables both in the present and in the future. The aforementioned characteristics converts the MPC in one of the preferred control methods for this kind of networks [9], [10].

Another interesting characteristic of this algorithm is that it allows to mathematically subdivide the overall system into smaller sections, usually known as "subsystems" or "Nodes" of the grid [11]. Each of these Nodes can be separately modelled as they have, for the most part, their own constraints and needs. The terms "large-scale system" and

© 2021 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 October 2021 doi:10.20944/preprints202110.0457.v1

"Nodes" are relative to each type of application. Thus, a large-scale system is essentially one that can be divided into subsystems and generally involves a high computational impact to directly implement its control mechanisms [12]. Depending on the type of modelling and its complexity, even a simple house can be considered a large-scale system [13]. However, thanks to state-space modelling methods (as Energy Hub methods [14]), large physical systems can be modelled in a simplified way. This fact, coupled with the robustness of MPCs against the errors introduced by the differences between models and real systems, and the simplicity of modelling large systems by dividing them into their respective Nodes, ensures that the mathematical modelling process is affordable, simple and efficient [15].

Once a system is modelled as the aggregate of its Nodes, it is a common practice to try to control the Nodes individually by applying an MPC agent to each of them, instead of controlling the whole network with a single control system, establishing or not communications between the different agents depending on the MPC algorithm [16]. Historically, the use of a single MPC for controlling the large-scale system has been avoided because the available computational processing power was not enough [17].

Alternatively, large-scale networks control problems used to be resolved by the subdivision of the control problem into several ones (usually one per node of the grid). In such scenario, it is always intended that the solution obtained for the whole network is approximately the same as the one that would have been obtained if only one MPC had been used to control the whole system. This is because the optimal, most robust and convergent solution is the one generated with a single controller [15].

Fortunately, the evolution of technology has led to an exponential increase in the processing power of computers, opening new ways towards the control of large-scale systems. The authors of this paper have seen in the evolution of technology the opportunity to improve one of the most historically complicated cases of control in large-scale systems: "flexibility mechanisms" (also called "coupled constraints" or "aggregate terms").

A "flexibility mechanism" is the need for several Nodes in a large-scale system to compensate or comply with constraints together [18], [19]. In this way, a set of variables from different Nodes in a network are made to participate cooperatively to meet a given constraint. An example of this would be to prevent the sum of certain variables from being greater than a value. The aggregate terms would be used to ensure, for example, that the combined electricity consumption of several houses does not exceed a value, that the flow rate of a specific liquid at several entry points of a chemical plant is not less than a value, that the set of several electricity generators provides at least a sum of energy, etc.

Since large-scale systems control problems have historically been subdivided into different nodes, the constraints of the flexibility mechanisms have also been heuristically subdivided so far. Consequently, each Node fulfilled part of the constraint independently. This approach implies a drastic decrease in the efficiency of control systems in terms of flexibility mechanisms, as the authors show in 5.1. In such a context, the authors propose in this paper an alternative mathematical method for the automatic and optimal calculation of the distribution of the flexibility mechanisms among all the Nodes of the grid. This method takes advantage of the benefits of Node modelling and does not negatively affect the convergence of the obtained solution while ensuring an optimal solution scope for the whole network.

To illustrate the effects and results obtained by applying this mathematical method, the rest of this paper is organised as follows. First, section 2 explains in more depth what a Model Predictive Control is. Later, section 4 shows the use case used in section 5 to exemplify the comparison between the traditional distribution of the flexibility mechanism using heuristic methods and the automated distribution method designed by the authors, which is shown in depth in section 3. Finally, section 7 concludes this work by listing the conclusions reached.

2. Model Predictive Control

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 October 2021 doi:10.20944/preprints202110.0457.v1

Model Predictive Control (MPC) is an advanced process control method that has been used in large-scale systems since the 80s [20]. MPCs are widely used in the industry since they allow to take into account both the current and also the future system status in order to satisfy the system constraints and control objectives [21]. This is achieved by minimising an objective function, which allows to define the cost associated to the use of a certain variables, throughout a prediction time horizon [22].

At each instant, the set of future control values is calculated by following the next criteria:

minimise	an objective function (also called "cost function") in terms of actions over the prediction horizon.
subject to	dynamics of the system over the prediction horizon; constraints of the system; measurement of the initial state of the system at the beginning of the current control instant.

Therefore, the MPC is based on an iterative optimisation. At instant k, the information about the current state of the system is recieved, and a control strategy that minimises the cost is calculated (using a mathematical solver) for a time range from the current instant (k) to a future instant ($k + H$). Specifically, a solution that emerges from the current state and guides the system through a time range $[k, k + H]$ is found (always complying with objective function cost minimisation and constraints). Then, only the next step (that correspond to $k + 1$) of the control strategy is implemented. Then, when the next instant $k + 1$ arrives, the process starts again.

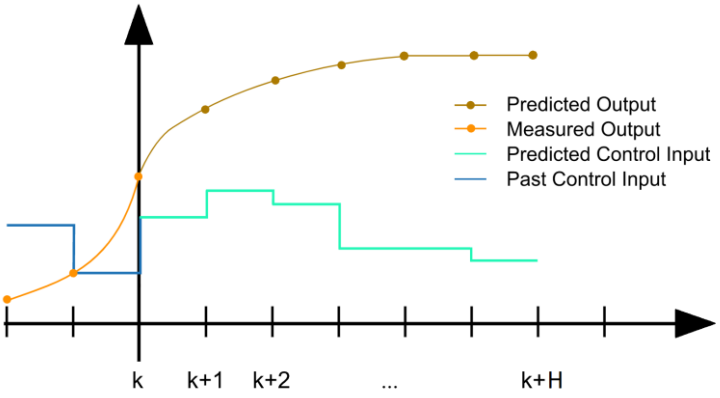

Figure 1: MPC scheme

MPC is one of the most widely used control systems for controlling large-scale processes. In general, the system model is subdivided into subsystems (also called "Nodes") to be applied to such processes. In this way, a complex system can be defined by modelling the Nodes that define the overall system and the relationship of each one with each other [20].

As mentioned above, the use of a single MPC which governs the entire large-scale system has been avoided due to the lack of available computational resources. On the contrary, it was preferred to design an MPC for each Node in the system. These MPCs, which may or may not communicate with each other, tried to obtain particular solutions that, as a whole, would result in a behaviour similar to the one obtained if a single MPC had been used [13]. Nevertheless, the communication mechanisms led to the emergence of slight delays and desynchronisation between different nodes

Due to the desynchronisation between the MPCs of the different Nodes, the flexibility mechanisms could not be solved efficiently [17]. In fact, in large-scale systems with several MPCs, a common practice is to deny the communication between the system's

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 October 2021 doi:10.20944/preprints202110.0457.v1

Nodes so that each Node's MPC works independently (an algorithm known as Decentralised MPC) [23]. Therefore, in order to comply with shared constraints between Nodes, this constraint needs to be subdivided heuristically so each Node had to comply with part of it separately.

Nevertheless, the computational capacity available nowadays has experienced a huge increase [24] and, for this reason, the authors have proposed and designed an MPC algorithm that optimises the whole system. In such a way, it obtains the greatest efficiency in the use of flexibility mechanisms with convergent and optimal solution without heuristically subdividing the shared constraints.

3. Flexibility mechanism with automatic methods: Centralised MPC

As mentioned above, this paper intends to show an MPC controller capable of working with flexibility mechanisms. This means that a set of variables from different Nodes of a network must participate together to satisfy a specific constraint. In this document, this constraint is exemplified by ensuring that the sum of the inputs of the various Nodes in the system is not greater than a specific value. To such an end, the authors propose to use a Centralised MPC control.

The Centralised MPC consists of an unique and sophisticated control agent that controls the whole large-scale system by itself. This Agent must take into account the state variables, inputs, outputs, constants, constraints, perturbations and dynamics of the all the elements of the network while satisfies a single global objective function for the entire system. Since this MPC knows every piece of information of every node of the grid, it ensures the convergence and the optimality (global optimum) of the solution [8]. However, some past literature [20] indicated that the disadvantages of a Centralised MPC include the difficulty of modelling the entire system, the large computational capacity required to have small runtimes, and the complication of isolating parts of the system in case one part fails.

Nevertheless, the authors have followed some methodologies that reduce the impact of the aforementioned disadvantages. First, the system has been modelled as an aggrupation of different and independent state-space models. Accordingly, the system can be easily modelled and the Nodes can be rapidly isolated if necessary. Furthermore, the computational requirements of Centralised MPC can be handled nowadays, in most cases, since the technology available today has exponentially increased its computational capacity during the last years [24].

The following figure shows a large-scale system composed of N parts, also called "Nodes", where each node has a set of inputs $u_i(t)$, state variables $x_i(t)$ and outputs $y_i(t)$ (Figure 2).

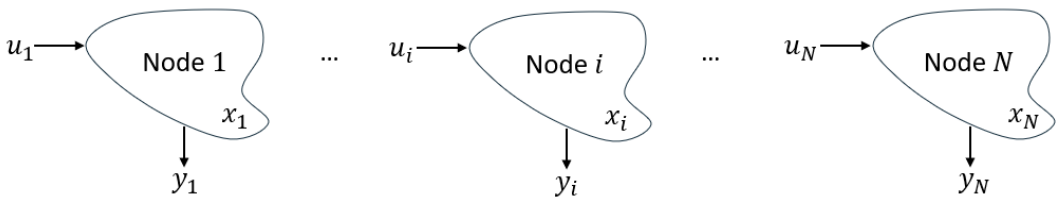

Figure 2. General network of N nodes

Where:

$$x_i = [x_{i,1}, \dots, x_{i,j}, \dots, x_{i,\alpha}]^T \quad (1)$$

$$u_i = [u_{i,1}, \dots, u_{i,j}, \dots, u_{i,\beta}]^T \quad (2)$$

$$y_i = [y_{i,1}, \dots, y_{i,j}, \dots, y_{i,\gamma}]^T \quad (3)$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 October 2021 doi:10.20944/preprints202110.0457.v1

Being i the number of nodes, α the number of states of the node, β the number of inputs of the node and γ the number of outputs of the node.

In general, the dynamics of each node is given by the continuous, non-linear and time-varying description:

$$\dot{x}_i\ (t) = f_i\big(x_i(t), u_i(t)\big) \tag{4}$$

$$y_i\ (t) = g_i\big(x_i(t), u_i(t)\big) \tag{5}$$

In a particular way, although it does not detract from the generality of the reasoning, each node can be modelled by a discrete, linear, time-invariant description with a state-space model [15]:

$$x_i(k+1) = A_i \cdot x_i(k) + B_i \cdot u_i(k) \tag{6}$$

$$y_i\ (k) = C_i \cdot x_i(k) \tag{7}$$

Where:

$$A_i = \begin{bmatrix} a^i_{1,1} & \cdots & a^i_{1,\alpha} \\ \vdots & \ddots & \vdots \\ a^i_{\alpha,1} & \cdots & a^i_{\alpha,\alpha} \end{bmatrix} \tag{8}$$

$$B_i = \begin{bmatrix} b^i_{1,1} & \cdots & b^i_{1,\beta} \\ \vdots & \ddots & \vdots \\ b^i_{\beta,1} & \cdots & b^i_{\beta,\beta} \end{bmatrix} \tag{9}$$

$$C_i = \begin{bmatrix} c^i_{1,1} & \cdots & c^i_{1,\gamma} \\ \vdots & \ddots & \vdots \\ c^i_{\gamma,1} & \cdots & c^i_{\gamma,\gamma} \end{bmatrix} \tag{10}$$

The local control problem for each node i can be written as:

$$\min_{\tilde{x}_i(k+1),\tilde{y}_i(k),\tilde{u}_i(k)} \phi_{local,i}\big(\tilde{x}_i(k+1), \tilde{u}_i(k), \tilde{y}_i(k)\big) \tag{11}$$

$$\begin{aligned} subject\ to \quad & x_i(k+1+l) = A_i x_i(k+l) + B_i u_i(k+l) \\ & y_i(k+l) = C_i x_i(k+l) \end{aligned} \tag{12}$$

Where k is the execution or simulation instant, l is the prediction window instant and:

$$\tilde{x}_i(k+1) = [x_i^T(k), \dots, x_i^T(k+H)]^T \tag{13}$$

$$\tilde{u}_i(k) = [u_i^T(k), \dots, u_i^T(k+H-1)]^T \tag{14}$$

$$y_i(k) = [y_i^T(k), \dots, y_i^T(k+H-1)]^T \tag{15}$$

$$x_i(k+1+l), u_i(k+l), y_i(k+l)\ \in\ \Omega_{x_i}, \Omega_{u_i}, \Omega_{y_i} \tag{16}$$

$$l = [0,1, \dots, H-1] \tag{17}$$

$$k \in N \tag{18}$$

Being H the control horizon.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 October 2021 doi:10.20944/preprints202110.0457.v1

Thanks to this mathematical description, a Centralised MPC can be defined for a large-scale system. Specifically, a Centralised MPC applied to a large-scale system of N nodes with a flexibility mechanism can be represented as follows (**Figure 3**).

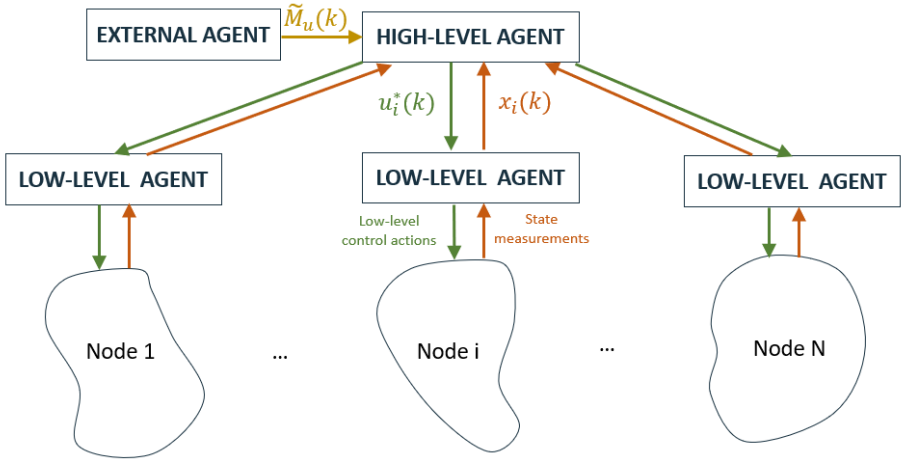

Figure 3: Generic Centralised MPC diagram

As can be seen in the figure, an External Agent defines the limits to be considered for the flexibility mechanism. In the case of the Centralised MPC, this External Agent only would have to define one global objective that all the nodes at the same time must satisfy. These limits are directly related to the variable $\tilde{M}_u(k)$ of the scheme which will be explained later in this section. This diagram also shows a High-Level Agent, which is the one that contains the Centralised MPC and performs the calculation of the optimal control actions in order to solve the optimisation problem. Afterwards, it transfers the optimisation solution reached to the individual agents (or Low-Level Agents) whose only responsibilities are applying the solutions given by the High-Level Agent, taking measurements about the current state of its node (under the vector $x_i(k)$), and send this information back to the High-Level Agent to begin with the next iteration. It should be noted that although the High-Level Agent computes the solution for all the variables involved in the optimisation problem and for the whole prediction horizon, it only sends those that are manipulated and adjustable (under the vector $u_i^*(k)$).

Regarding the mathematical definition, the control problem of a large-scale system of N interconnected nodes can be seen as an aggregation of the local control problems plus the constraints of the flexibility mechanisms. This controlled aggregation is, after all, a Centralised MPC. Its equations can be extrapolated from Equations (6)-(18) and are as follows.

$$\min_{\tilde{x}_i(k+1),\tilde{y}_i(k),\tilde{u}_i(k)} \sum_{i=1}^{N} \phi_{local,i}\left(\tilde{x}_i(k+1),\tilde{u}_i(k),\tilde{y}_i(k)\right) \tag{19}$$

$$\begin{aligned} subject\ to \quad & \boldsymbol{X}(k+1+l) = \boldsymbol{A}\boldsymbol{X}(k+l) + \boldsymbol{B}\boldsymbol{U}(k+l) \\ & \boldsymbol{Y}(k+l) = \boldsymbol{C}\boldsymbol{X}(k+l) \\ & \boldsymbol{K}[\cdot]\cdot\boldsymbol{U}(k+l) \le \tilde{M}_u(k+l) \end{aligned} \tag{20}$$

Where:

$$\boldsymbol{A} = \begin{bmatrix} A_1 & & & \\ & A_2 & & \\ & & \ddots & \\ & & & A_N \end{bmatrix} \tag{21}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 October 2021 doi:10.20944/preprints202110.0457.v1

$$\boldsymbol{B} = \begin{bmatrix} B_1 & & & \\ & B_2 & & \\ & & \ddots & \\ & & & B_N \end{bmatrix} \tag{22}$$

$$\boldsymbol{C} = \begin{bmatrix} C_1 & & & \\ & C_2 & & \\ & & \ddots & \\ & & & C_N \end{bmatrix} \tag{23}$$

$$\boldsymbol{X}(k+1) = [x_1^T(k+1) \quad \cdots \quad x_N^T(k+1)]^T \tag{24}$$

$$\boldsymbol{U}(k) = [u_1^T(k) \quad \cdots \quad u_N^T(k)]^T \tag{25}$$

$$\boldsymbol{Y}(k) = [y_1^T(k) \quad \cdots \quad y_N^T(k)]^T \tag{26}$$

$$\widetilde{M}_u(k+l) \in R^+ \tag{27}$$

About these equations, it is essential to note that:

1. Since $k+1>0$, it is essential to initialise $\boldsymbol{X}$ such that:

$$\boldsymbol{X}(0) = [x_1^T(0) \quad \cdots \quad x_N^T(0)]^T \tag{28}$$

2. The flexibility mechanism has been added by the last constraint of Equation (20):

$$\boldsymbol{K}[\cdot] \cdot \boldsymbol{U}(k+l) \leq \widetilde{M}_u(k+l) \tag{29}$$

With this inequation, the sum of specific inputs must be less than a value at each instant. This sum is obtained through the vector $\boldsymbol{K}[\cdot]$. This vector, made up of a binary value for each of the system's inputs, makes it possible to determine which inputs must follow the restriction (associating a "1" in their position).

Thus, for example, in a system of two Nodes with a total of four inputs where it is imposed that all but the third one must comply with the restriction at a given time:

$$\left\{ [1 \quad 1 \quad 0 \quad 1] \begin{bmatrix} u_{11} \\ u_{12} \\ u_{21} \\ u_{22} \end{bmatrix} \leq \widetilde{M}_u \right\} \rightarrow \{u_{11} + u_{12} + u_{22} \leq \widetilde{M}_u\} \tag{30}$$

Where u_{ij} is the input j of the Node i.

These equations define the general mathematical formulation of a Centralised MPC applied to a large-scale system.

As a final step, it is essential to explain how the MPC works. As explained above, the MPC consists of an iterative process where at each execution instant k, the optimal control solution is determined by the following steps:

1. Feed-back the values of the state variables $\boldsymbol{X}$ calculated in the previous instant. $\widetilde{M}_u$ is also received for the current instant. Both data are updated in the matrices.
2. Calculate the Centralised MPC problem according to the Equations (19)-(29).
3. Implement the solution.
4. Return to step 1.

4. Use case

As mentioned above, the main objective of the proposed approach is to get several parts of a large-scale system to comply with shared constraints. That is, to get these parts, which are modelled separately, to cooperate to meet the overall constraints together. To

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 October 2021 doi:10.20944/preprints202110.0457.v1

this end, the authors propose a use case applied to the generalised large-scale system modelled by its Nodes using Equations (6)-(18). The results achieved after the simulation of this academic use case will be discussed in the upcoming section to analyse the behaviour and performance of the solution proposed in this paper.

Specifically, the academic use case included in this document relies on a four-node network where each node has a defined number of variables. In concrete terms, the model of each node corresponds to the model of Node i shown in the following image (Figure 4).

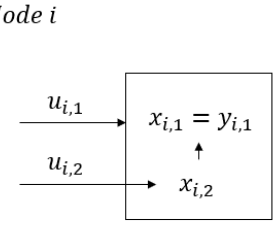

Figure 4: Diagram and variables of the nodes of the network defined in the use case

Each node has two inputs ($u_{i,1}$ and $u_{i,2}$), two states ($x_{i,1}$ and $x_{i,2}$) and one output ($y_{i,1}$). In the figure below, the scheme of the Centralised MPC for the use case is depicted.

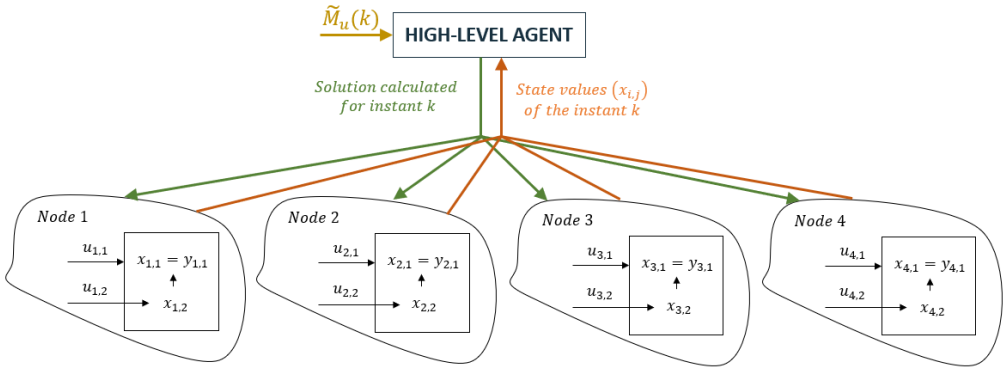

Figure 5: Centralised MPC applied to the use case

The model of each of the four nodes of the use case corresponds to a particular state space, as shown in Equations (6)-(10):

$$Node\ 1 \begin{cases} \underbrace{\begin{bmatrix} x_{1,1}(k+1) \\ x_{1,2}(k+1) \end{bmatrix}}_{x_1(k+1)} = \underbrace{\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}}_{A_1} \underbrace{\begin{bmatrix} x_{1,1}(k) \\ x_{1,2}(k) \end{bmatrix}}_{x_1(k)} + \underbrace{\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}}_{B_1} \underbrace{\begin{bmatrix} u_{1,1}(k) \\ u_{1,2}(k) \end{bmatrix}}_{u_1(k)} \\ \underbrace{\begin{bmatrix} y_{1,1}(k) \end{bmatrix}}_{y_1} = \underbrace{\begin{bmatrix} 1 & 0 \end{bmatrix}}_{C_1} \underbrace{\begin{bmatrix} x_{1,1}(k) \\ x_{1,2}(k) \end{bmatrix}}_{x_1(k)} \end{cases} \quad (31)$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 October 2021 doi:10.20944/preprints202110.0457.v1

$$Node\ 2 \begin{cases} \underbrace{\begin{bmatrix} x_{2,1}(k+1) \\ x_{2,2}(k+1) \end{bmatrix}}_{x_2(k+1)} = \underbrace{\begin{bmatrix} 1 & 1 \\ 0 & 0.7 \end{bmatrix}}_{A_2} \underbrace{\begin{bmatrix} x_{2,1}(k) \\ x_{2,2}(k) \end{bmatrix}}_{x_2(k)} + \underbrace{\begin{bmatrix} -0.2 & 0 \\ 0 & -0.2 \end{bmatrix}}_{B_2} \underbrace{\begin{bmatrix} u_{2,1}(k) \\ u_{2,2}(k) \end{bmatrix}}_{u_2(k)} \\ \underbrace{\begin{bmatrix} y_{2,1}(k) \end{bmatrix}}_{y_2} = \underbrace{\begin{bmatrix} 1 & 0 \end{bmatrix}}_{C_2} \underbrace{\begin{bmatrix} x_{2,1}(k) \\ x_{2,2}(k) \end{bmatrix}}_{x_2(k)} \end{cases} \quad (32)$$

$$Node\ 3 \begin{cases} \underbrace{\begin{bmatrix} x_{3,1}(k+1) \\ x_{3,2}(k+1) \end{bmatrix}}_{x_3(k+1)} = \underbrace{\begin{bmatrix} 1 & 0.9 \\ 0 & 0.8 \end{bmatrix}}_{A_3} \underbrace{\begin{bmatrix} x_{3,1}(k) \\ x_{3,2}(k) \end{bmatrix}}_{x_3(k)} + \underbrace{\begin{bmatrix} -0.8 & 0 \\ 0 & -0.8 \end{bmatrix}}_{B_3} \underbrace{\begin{bmatrix} u_{3,1}(k) \\ u_{3,2}(k) \end{bmatrix}}_{u_3(k)} \\ \underbrace{\begin{bmatrix} y_{3,1}(k) \end{bmatrix}}_{y_3} = \underbrace{\begin{bmatrix} 1 & 0 \end{bmatrix}}_{C_3} \underbrace{\begin{bmatrix} x_{3,1}(k) \\ x_{3,2}(k) \end{bmatrix}}_{x_3(k)} \end{cases} \quad (33)$$

$$Node\ 4 \begin{cases} \underbrace{\begin{bmatrix} x_{4,1}(k+1) \\ x_{4,2}(k+1) \end{bmatrix}}_{x_4(k+1)} = \underbrace{\begin{bmatrix} 1 & 0.8 \\ 0 & 0.7 \end{bmatrix}}_{A_4} \underbrace{\begin{bmatrix} x_{4,1}(k) \\ x_{4,2}(k) \end{bmatrix}}_{x_4(k)} + \underbrace{\begin{bmatrix} -1.7 & 0 \\ 0 & -1.7 \end{bmatrix}}_{B_4} \underbrace{\begin{bmatrix} u_{4,1}(k) \\ u_{4,2}(k) \end{bmatrix}}_{u_4(k)} \\ \underbrace{\begin{bmatrix} y_{4,1}(k) \end{bmatrix}}_{y_4} = \underbrace{\begin{bmatrix} 1 & 0 \end{bmatrix}}_{C_4} \underbrace{\begin{bmatrix} x_{4,1}(k) \\ x_{4,2}(k) \end{bmatrix}}_{x_4(k)} \end{cases} \quad (34)$$

In these matrix systems, each of the inputs of Node i (u_i variables) is directly associated with a state (x_i variables). Thus, the use of $u_{i,1}$ directly impacts $x_{i,1}$ and the use of $u_{i,2}$ directly impacts $x_{i,2}$. Also, since $x_{i,1}$ is dependent on $x_{i,2}$, the use of $x_{i,2}$ impacts $x_{i,1}$ and, consequently, the use of $u_{i,2}$ indirectly impacts $x_{i,1}$ (Figure 4). This dependency between states of the same node is a complex situation to control. However, it also allows showing the behaviour of the control proposed in this paper visually and straightforwardly in limit situations. The authors have therefore considered this use case as the best one for the analysis.

Applying the use case to the Centralised MPC, it can be ensured that the objective function (Equation (19)) satisfies the following equivalence:

$$\begin{aligned} &\min_{\tilde{x}_i(k+1),\tilde{y}_i(k),\tilde{u}_i(k)} \sum_{i=1}^{N} \phi_{local,i}\left(\tilde{x}_i(k+1), \tilde{u}_i(k), \tilde{y}_i(k)\right) = \\ &= \min_{\tilde{x}_i(k+1),\tilde{y}_i(k),\tilde{u}_i(k)} \sum_{i=1}^{N} \begin{bmatrix} \boldsymbol{X}(k+1+l) \\ \boldsymbol{U}(k+l) \\ \boldsymbol{Y}(k+l) \end{bmatrix}^T \cdot \boldsymbol{Q} \cdot \begin{bmatrix} \boldsymbol{X}(k+1+l) \\ \boldsymbol{U}(k+l) \\ \boldsymbol{Y}(k+l) \end{bmatrix} + \boldsymbol{f}^T \cdot \begin{bmatrix} \boldsymbol{X}(k+1+l) \\ \boldsymbol{U}(k+l) \\ \boldsymbol{Y}(k+l) \end{bmatrix} \end{aligned} \quad (35)$$

Where $\boldsymbol{Q}$ is the matrix that contains the quadratic cost of each variable, and $\boldsymbol{f}$ is the matrix that contains the linear cost of each variable.

The use case data of Equations (31)-(34) can be implemented in the Equations (19)-(29),(35) in such a way that:

- Model:

$$\boldsymbol{A} = \begin{bmatrix} A_1 & & & \\ & A_2 & & \\ & & A_3 & \\ & & & A_4 \end{bmatrix} \quad \boldsymbol{B} = \begin{bmatrix} B_1 & & & \\ & B_2 & & \\ & & B_3 & \\ & & & B_4 \end{bmatrix} \quad \boldsymbol{C} = \begin{bmatrix} C_1 & & & \\ & C_2 & & \\ & & C_3 & \\ & & & C_4 \end{bmatrix} \quad (36)$$

$$\boldsymbol{X} = [x_{11} \quad x_{12} \quad x_{21} \quad x_{22} \quad x_{31} \quad x_{32} \quad x_{41} \quad x_{42}]^T$$
$$\boldsymbol{U} = [u_{11} \quad u_{12} \quad u_{21} \quad u_{22} \quad u_{31} \quad u_{32} \quad u_{41} \quad u_{42}]^T$$
$$\boldsymbol{Y} = [y_{11} \quad y_{21} \quad y_{31} \quad y_{41}]^T$$

- Objective function:

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 October 2021 doi:10.20944/preprints202110.0457.v1

$$\boldsymbol{Q} = \begin{bmatrix} Q_X & & \\ & Q_U & \\ & & Q_Y \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{8x8} & & \\ & \mathbf{0}_{8x8} & \\ & & \mathbf{0}_{4x4} \end{bmatrix}; \quad \boldsymbol{f} = \begin{bmatrix} f_X \\ f_U \\ f_Y \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{8x1} \\ \mathbf{0}_{8x1} \\ \mathbf{1}_{4x1} \end{bmatrix} \tag{37}$$

Where: $\mathbf{0}_{hxj}$ and $\mathbf{1}_{hxj}$ are matrixes filled with zeros or ones with order hxj.

- Flexibility mechanism:

$$\widetilde{M}_u(k+l) = 4\,, \forall k, \forall l = [0,1,\dots,H-1] \\ \boldsymbol{K}[\cdot] = \mathbf{1}_{1x8} \tag{38}$$

- Initial condition ($k = 0$):

$$\boldsymbol{X}(0) = [8 \quad 7 \quad 6 \quad 5 \quad 4 \quad 3 \quad 4 \quad 5]^T \tag{39}$$

- Space of variables:

$$u_{ij}(k+l), x_{ij}(k+l+1), y_{ij}(k+l) \geq 0,0,0 \tag{40}$$

- Prediction horizon:

$$H = 5 \tag{41}$$

Using this information, the Centralised MPC control problem is particularised for the use case, and a simulation can be performed.

5. Benchmark

As highlighted above, one of the simplest and widely adopted methods to get multiple Nodes in a network to comply with an overall constraint is to heuristically distribute this constraint between all the Nodes of the grid. That way, each of the Nodes must independently satisfy their respective assigned part of the overall constraint. Thus, in order to establish a benchmark test for the proposed solution, an heuristic method is proposed.

5.1. Flexibility mechanisms using heuristic methods: Decentralised MPC

One of the ways to control a large-scale system from the individual control of its nodes is the so-called "Decentralised MPC". This method is based on modelling and controlling each node of a system independently of the others. Thus, instead of controlling a complete system, a small controller is made for each node. Historically, this method has been widely used because of its lower computational demand. Nowadays, alternative modelling methods (such as the Centralised MPC proposed by the authors) allow the whole network to be modelled from the model of each of its nodes. However, Decentralised MPC is still used today [25].

The Decentralised MPC is based on subdividing the whole system into small subsystems. In this way, the large-scale system becomes a set of small-scale systems. Each of these subsystems has its own MPC agent, which is responsible for fulfilling the constraints, dynamics and objective function of the model of its particular subsystem. Each of these agents ignores the rest of the agents, i.e., they compute their optimal solution independently [26]. In other words, there is no communication between agents, neither any attempt to reach a global optimum. Instead, each subsystem reaches a local optimum and, thus, an approximation of a global optimum is found. Such an approximation is usually not very close to the global optimum because, given the lack of interactions between the subsystems, the sum of the optimal solutions is not equal to the global optimum.

The absence of a guarantee for global optimum is not the only disadvantage that the Decentralised MPC presents when applied to a large-scale system. If several nodes are intended to meet a constraint jointly, this constraint will have to be heuristically subdivided. This is because the lack of communication prevents them from agreeing on how to deal with the constraint. This does not mean that the restriction will not be met, but it does mean that, if it is met, it may not be done as effectively, quickly and efficiently as possible.

A Decentralised MPC applied to a large-scale system of N nodes with a flexibility mechanism can be represented as follows (**Figure 6**).

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 October 2021 doi:10.20944/preprints202110.0457.v1

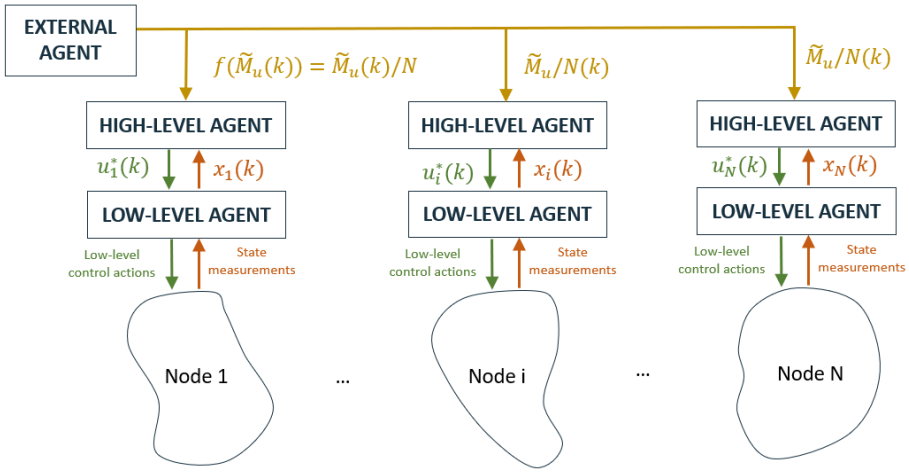

Figure 6: Generic Decentralised MPC diagram

In the specific case of the Decentralised MPC of the diagram above (**Figure 6**), specific characteristics can be highlighted:

- The relationship of a High-Level Agent to its Low-Level Agent is the same as in the Centralised MPC. Each High-Level Agent contains an MPC that performs the control, while the particular Low-Level Agent is in charge of transforming the High-Level Agent's solutions into actual actions on its subsystem, also taking data measurements from the Node.
- Since the Nodes do not communicate with each other, the only way to try to fulfil the flexibility mechanisms is to distribute them heuristically among the agents. One way to do this is to divide it equally among all the agents, making each of them must comply: $\tilde{M}_u(k)/_N$. This distribution could be done in other ways, but the responsibility of this calculation will always rely on the external agent while the solution proposed by the authors divide the effort among the nodes automatically and optimally.

Regarding the mathematical definition, each Node has an Agent with the same generic mathematical formulation for its internal MPC, which mainly corresponds to the Node presented in Equations (6)-(18). To these equations is added a term that heuristically subdivides a flexibility mechanism between the nodes (Equation (45)). This MPC, independent of the others, is particularised by the dynamics and model of its particular Node. Thus, i is the number of the modelled node, where $i \in [1,2,\dots,N]$, being N the number of nodes in the system.

$$\min_{\tilde{x}_i(k+1),\tilde{y}_i(k),\tilde{u}_i(k)} \sum_{i=1}^{N} \phi_{local,i}\left(\tilde{x}_i(k+1), \tilde{u}_i(k), \tilde{y}_i(k)\right) \tag{42}$$

$$\begin{aligned} subject\ to \quad & x_i(k+1+l) = A_i x_i(k+l) + B_i u_i(k+l) \\ & y_i(k+l) = C_i x_i(k+l) \\ & K_i[\cdot] \cdot u_i(k+l) \leq \frac{\tilde{M}_u(k+l)}{N} \end{aligned} \tag{43}$$

About these equations, it is essential to note that:

1. A_i, B_i and C_i, the matrixes of the model of the Node, have the exact definition than in Equations (8)-(10).
2. Since $k+1 > 0$, it is essential to initialise $x_i(0)$ such that:

$$x_i(0) = \begin{bmatrix} x_{i,1}^T(0) & \cdots & x_{i,\alpha}^T(0) \end{bmatrix}^T \tag{44}$$

3. The flexibility mechanism has been added by the last constraint of Equation (43):

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 October 2021 doi:10.20944/preprints202110.0457.v1

$$K_i[\cdot] \cdot u_i(k+l) \leq \frac{\widetilde{M}_u(k+l)}{N} \tag{45}$$

With this inequation, the sum of specific inputs of the Node must be less than a value at each instant. This sum is obtained through the vector $K_i[\cdot]$. This vector, made up of a binary value for each Node's inputs, makes it possible to determine which inputs must follow the restriction (associating a "1" in their position).

Thus, for example, in a Node with two inputs where it is imposed that both must comply with the restriction at a given time:

$$\left\{ \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} u_{11} \\ u_{12} \end{bmatrix} \leq \frac{\widetilde{M}_u}{N} \right\} \rightarrow \left\{ u_{11} + u_{12} \leq \frac{\widetilde{M}_u}{N} \right\} \tag{46}$$

Where u_{ij} is the input j of the Node i.

Because each MPC agent only needs to fulfil a portion of the flexibility mechanism and there is no communication between subsystems, even if an agent can support another agent to fulfil its flexibility mechanism, it will not do so. However, the local optimum in each Node is still guaranteed.

These equations define the general mathematical formulation of a Decentralised MPC utilised to a large-scale system. As a final step, it is essential to consider how the Decentralised MPC must be executed. At each execution instant, each Agent follows the following stages synchronously and in parallel, achieving their optimal local solution and using them as an approximation of an optimal global solution:

1. Feed-back the values of the state variables x_i calculated in the previous instant. $\widetilde{M}_u/_N$ is also received for the current instant. Both data are updated in the matrices.
2. Calculate the control problem of the Agent according to the equations shown above.
3. Implement the solution.
4. Return to step 1.

5.2. Decentralised MPC applied to the use case

Applying the use case defined in the previous section, it can be ensured that the local objective function of a Node (Equation (42)) satisfies the following equivalence:

$$\begin{aligned} &\min_{\tilde{x}_i(k+1),\tilde{y}_i(k),\tilde{u}_i(k)} \phi_{local,i}\big(\tilde{x}_i(k+1),\tilde{u}_i(k),\tilde{y}_i(k)\big) = \\ &= \min_{\tilde{x}_i(k+1),\tilde{y}_i(k),\tilde{u}_i(k)} \begin{bmatrix} x_i(k+1+l) \\ u_i(k+l) \\ y_i(k+l) \end{bmatrix}^T \cdot Q_i \cdot \begin{bmatrix} x_i(k+1+l) \\ u_i(k+l) \\ y_i(k+l) \end{bmatrix} + f_i^T \cdot \begin{bmatrix} x_i(k+1+l) \\ u_i(k+l) \\ y_i(k+l) \end{bmatrix} \end{aligned} \tag{47}$$

Where Q_i is the matrix that contains the quadratic cost of each variable of the Node, and f_i is the matrix that contains the linear cost of each variable of the Node.

The use case data (Equations (31)-(34)) can be implemented in the Equations (42)-(44),(47) in such a way that:

- Model:

$$\begin{aligned} &Node\ 1 \rightarrow \{A_1 \quad B_1 \quad C_1\}, \quad x_1 = \begin{bmatrix} x_{11} \\ x_{12} \end{bmatrix}, \quad u_1 = \begin{bmatrix} u_{11} \\ u_{12} \end{bmatrix}, \quad y_1 = [y_{11}] \\ &Node\ 2 \rightarrow \{A_2 \quad B_2 \quad C_2\}, \quad x_2 = \begin{bmatrix} x_{21} \\ x_{22} \end{bmatrix}, \quad u_2 = \begin{bmatrix} u_{21} \\ u_{22} \end{bmatrix}, \quad y_2 = [y_{21}] \\ &Node\ 3 \rightarrow \{A_3 \quad B_3 \quad C_3\}, \quad x_3 = \begin{bmatrix} x_{31} \\ x_{32} \end{bmatrix}, \quad u_3 = \begin{bmatrix} u_{31} \\ u_{32} \end{bmatrix}, \quad y_3 = [y_{31}] \\ &Node\ 4 \rightarrow \{A_4 \quad B_4 \quad C_4\}, \quad x_4 = \begin{bmatrix} x_{41} \\ x_{42} \end{bmatrix}, \quad u_4 = \begin{bmatrix} u_{41} \\ u_{42} \end{bmatrix}, \quad y_4 = [y_{41}] \end{aligned} \tag{48}$$

- Objective function:

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 October 2021 doi:10.20944/preprints202110.0457.v1

$$Q_i = \begin{bmatrix} \mathbf{0}_{2x2} & & \\ & \mathbf{0}_{2x2} & \\ & & 0 \end{bmatrix}; \quad f_i = \begin{bmatrix} \mathbf{0}_{2x1} \\ \mathbf{0}_{2x1} \\ 1 \end{bmatrix}; \quad \forall i \tag{49}$$

Where: $\mathbf{0}_{hxj}$ is a matrix filled with zeros with order hxj.

- Flexibility mechanism:

$$\begin{gathered} \widetilde{M}_u(k+l) = 4\,, \forall k, \forall l = [0,1,\dots,H-1] \\ K_i[\cdot] = [1 \quad 1] \end{gathered} \tag{50}$$

- Initial conditions ($k = 0$):

$$\begin{gathered} x_1(0) = [8 \quad 7]^T \\ x_2(0) = [6 \quad 5]^T \\ x_3(0) = [4 \quad 3]^T \\ x_4(0) = [4 \quad 5]^T \end{gathered} \tag{51}$$

- Space of variables:

$$u_{ij}(k+l), x_{ij}(k+l+1), y_{ij}(k+l) \geq 0, 0, 0 \tag{52}$$

- Prediction horizon:

$$H = 5 \tag{53}$$

With this, the Decentralised MPC control problem is particularised for the use case. The diagram that describes the decentralised MPC particularised for the use case is the following one:

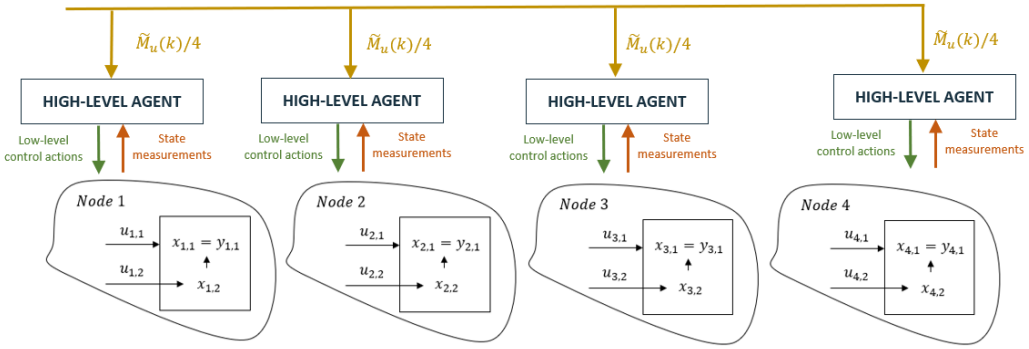

Figure 7: Decentralised MPC applied to the use case

6. Analysis of results

In this section, an overview and comparison of the results achieved by the two different architectures is performed. To such an end, a new Mathematical Framework, which contains both types of control algorithms (Centralised MPC and Decentralised MPC), have been implemented in Python. In addition, a simulation environment based on Jupyter Notebooks has also been deployed in order to carry out the simulations and generate the results presented and analysed in this section.

Figure 8, **Figure 9** and **Figure 10** show the results obtained from running the Centralised MPC with the use case.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 October 2021 doi:10.20944/preprints202110.0457.v1

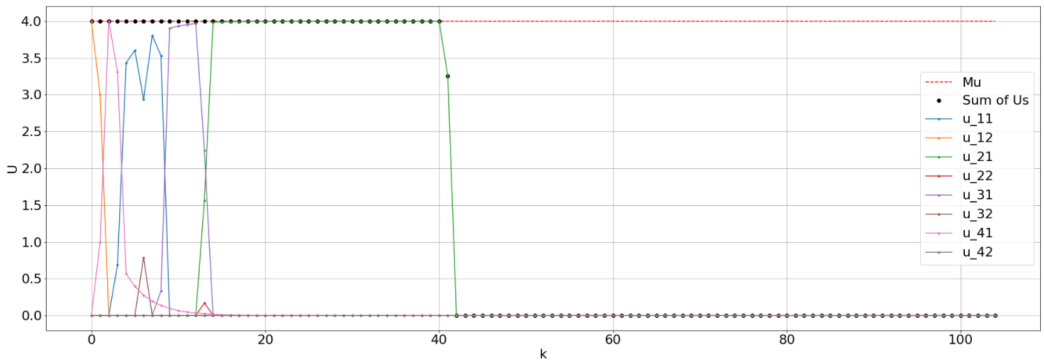

Figure 8: Solution for u-variables of the Centralised MPC

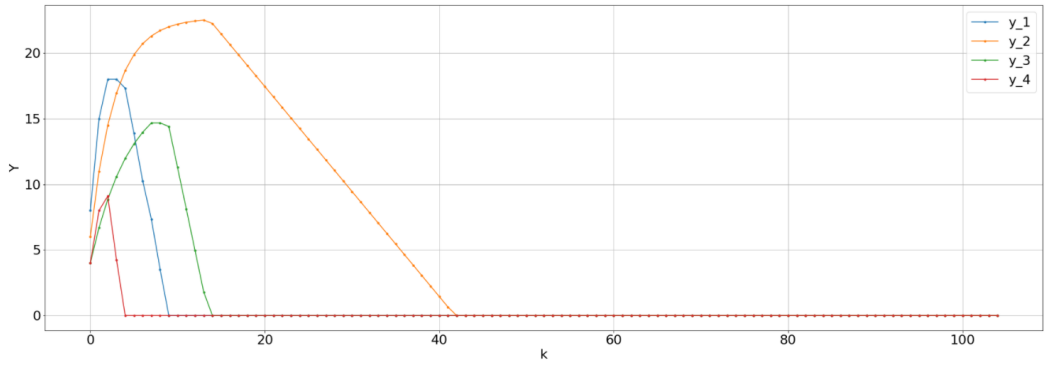

Figure 9: Solution for y-variables of the Centralised MPC

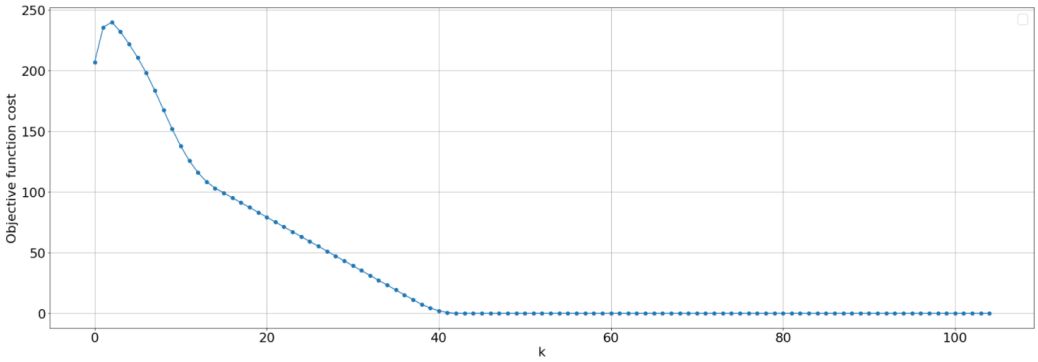

Figure 10: Objective function cost of the Centralised MPC

Figure 9 shows the value of the y_i variables (outputs) over a simulation of 105 instants. As can be extrapolated from Equation (35), the objective function of the Centralised MPC seeks to minimise the sum of the value of these variables in all the prediction window instants.

$$\begin{aligned} &\min_{\tilde{x}_i(k+1),\tilde{y}_i(k),\tilde{u}_i(k)} \begin{bmatrix} \boldsymbol{X}(k+1+l) \\ \boldsymbol{U}(k+l) \\ \boldsymbol{Y}(k+l) \end{bmatrix}^T \cdot \boldsymbol{Q} \cdot \begin{bmatrix} \boldsymbol{X}(k+1+l) \\ \boldsymbol{U}(k+l) \\ \boldsymbol{Y}(k+l) \end{bmatrix} + \boldsymbol{f}^T \cdot \begin{bmatrix} \boldsymbol{X}(k+1+l) \\ \boldsymbol{U}(k+l) \\ \boldsymbol{Y}(k+l) \end{bmatrix} \\ &= \min_{\tilde{x}_i(k+1),\tilde{y}_i(k),\tilde{u}_i(k)} \sum_{i=1}^{N} \begin{bmatrix} \mathbf{0}_{8x1} \\ \mathbf{0}_{8x1} \\ \mathbf{1}_{4x1} \end{bmatrix}^T \cdot \begin{bmatrix} \boldsymbol{X}(k+1+l) \\ \boldsymbol{U}(k+l) \\ \boldsymbol{Y}(k+l) \end{bmatrix} \\ &= \min_{\tilde{x}_i(k+1),\tilde{y}_i(k),\tilde{u}_i(k)} \sum_{i=1}^{N} \boldsymbol{Y}(k+l) \end{aligned} \tag{54}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 October 2021 doi:10.20944/preprints202110.0457.v1

As can be seen in **Figure 10**, this minimum is reached at simulation instant 42, which coincides with the moment when all y_i variables have reached the value 0 in **Figure 9**.

To reach this global minimum as quickly and efficiently as possible, the MPC decides to use the u_{ij} variables most effectively and optimally. The way to do that is to give them the highest value during all instants. This value is constrained by the flexibility mechanism, as seen in Equation (29).

Thus, the Centralised MPC will maximise the value of the u_{ij} variables by complying with this constraint. In this way, as can be seen in **Figure 8**, it will make the sum of the u_{ij} always worth $\widetilde{M}_u$ until instant 40, which will allow reaching the optimum at simulation instant 42. With this, the MPC has used all the possible values in the u_{ij} variables to reach the minimum value in the cost function as soon as possible.

The solution of this control problem is in accordance with the mathematical definition of Centralised MPC, which ensures convergence and global optimum. Moreover, the lowest value of the objective function is reached as soon as possible, which guarantees a shorter response time. It is important to note that, as already demonstrated, the most efficient and optimal fulfilment of the flexibility mechanisms is guaranteed.

On the other hand, the results of applying the Decentralised MPC in the use case are shown in **Figure 11**, **Figure 12**, **Figure 13** and **Figure 14**.

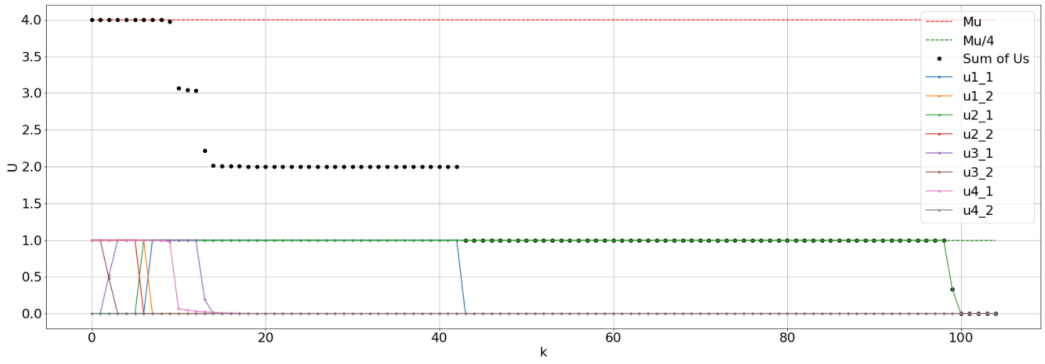

Figure 11: Solution for U-variables of the Decentralised MPC

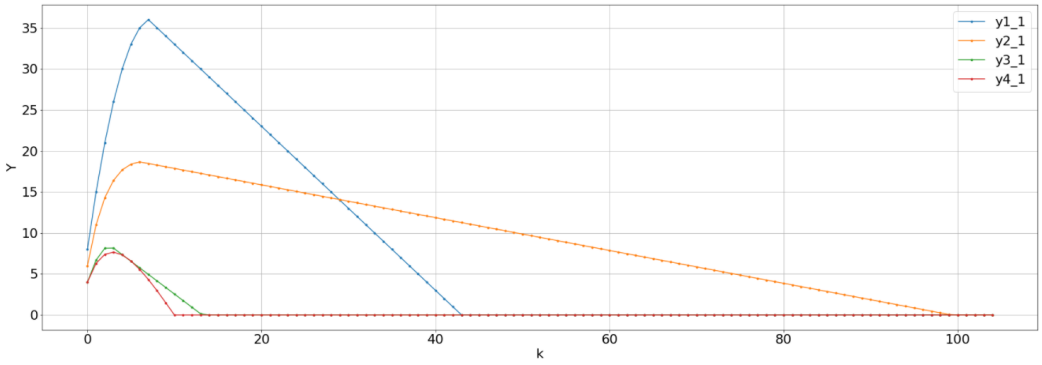

Figure 12: Solution for Y-variables of the Decentralised MPC

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 October 2021 doi:10.20944/preprints202110.0457.v1

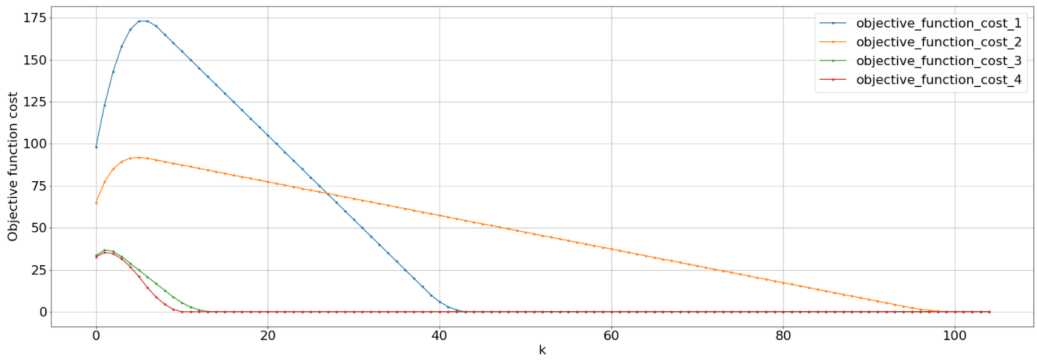

Figure 13: Local objective function costs of the Decentralised MPC

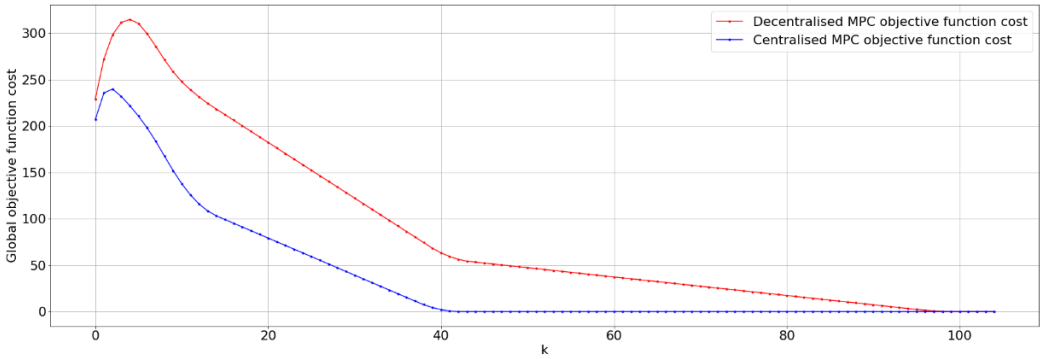

Figure 14: Sum of local objective function costs of the Decentralised MPC and objective function cost of Centralised MPC

As already explained above, the Decentralised MPC is based on the fact that each node executes an MPC independently from the others. Because of this, the flexibility mechanisms are arbitrarily divided. In this case, they have been divided equally, as seen in Equation (45).

This division of the compensating mechanisms affects the time it takes to reach the minimum value of the local objective functions, as shown in **Figure 13**.

As can be extrapolated from the Equation (47), the objective function of the Agents of the Decentralised MPC seek to minimise the value of the y_{ij} variables in all the prediction window instants.

$$\begin{aligned} \min_{\tilde{x}_i(k+1),\tilde{y}_i(k),\tilde{u}_i(k)} & \begin{bmatrix} x_i(k+1+l) \\ u_i(k+l) \\ y_i(k+l) \end{bmatrix}^T \cdot Q_i \cdot \begin{bmatrix} x_i(k+1+l) \\ u_i(k+l) \\ y_i(k+l) \end{bmatrix} + f_i^T \cdot \begin{bmatrix} x_i(k+1+l) \\ u_i(k+l) \\ y_i(k+l) \end{bmatrix} \\ &= \min_{\tilde{x}_i(k+1),\tilde{y}_i(k),\tilde{u}_i(k)} \begin{bmatrix} \mathbf{0}_{2x1} \\ \mathbf{0}_{2x1} \\ 1 \end{bmatrix}^T \cdot \begin{bmatrix} x_i(k+1+l) \\ u_i(k+l) \\ y_i(k+l) \end{bmatrix} \\ &= \min_{\tilde{x}_i(k+1),\tilde{y}_i(k),\tilde{u}_i(k)} y_i(k+l) \end{aligned} \tag{55}$$

The reason why the flexibility mechanism affects the execution time negatively in the Decentralised MPC is straightforward. As shown in **Figure 12**, each MPC of the Decentralised MPC works independently, so the value of y_{ij} decreases independently. This means that when a Node gets its y_{ij} to be equal to 0, it cannot concede any amount of its flexibility mechanism for another node to increase the value of its u_{ij} variables. This is easily seen in **Figure 11**, where it can be observed that only during the first instants the whole limit of the flexibility mechanism is used. In subsequent instants, even if a Node has already reached the minimum value, the total capacity of the mechanism will not be

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 October 2021 doi:10.20944/preprints202110.0457.v1

used again. Thus, it can be ensured that the Decentralised MPC does not use the flexibility mechanism most optimally.

It could be thought that a solution to take better advantage of the limits of the flexibility mechanism in this heuristic method is to avoid the continuous equal distribution of the flexibility mechanism. However, complementary methods for dynamic distribution of mechanisms (such as prediction or neural network systems) would be so difficult to make and costly to implement. To create them, it also would be necessary to know information about the global system when Decentralised MPC seeks just the opposite: deglobalisation. In fact, the best result for a dynamic distribution of flexibility mechanisms is actually the one obtained by the Centralised MPC, since the optimal distribution is ensured by the mathematics of the MPC.

Still, it is essential to note that the Decentralised MPC ensures convergence and local optimums, as shown. But it has not reached an efficient global optimum in this case, which confirms that it is not guaranteed that the set of local optimums always approximates a valid global optimum in all cases. It can be affirmed then that the Decentralised MPC reached a suboptimum solution. This is because the response time (the time it takes to approximate the global optimum) may be greater than desired. This effect occurs since the Decentralised MPC takes many instants to reach a global optimum (due to the inefficiency when applying flexibility mechanisms), demonstrating tiny efficiency and a poor response time. This can be seen in **Figure 14**, which shows how the sum of the values of the objective functions takes many simulation instants to reach zero.

Comparing the results of the Centralised MPC and the Distributed MPC, it can be noted from **Figure 14** that the Centralised MPC takes advantage of all the resources of the flexibility mechanism. In contrast, the Decentralised MPC takes the longest to reach the optimum. Another way to measure the speed with which the various algorithms have reached the global optimum is to measure the area of the curves in **Figure 14**. In the case of the Centralised MPC, this area reaches a value of 3925.4; in the Decentralised MPC, it reaches a value of 9205.6. This occurs since the outputs of the Centralised MPC decrease efficiently and quickly, getting the value of all of them to reach zero as soon as possible without breaking the constraints of the flexibility mechanism. On the other hand, the outputs in the Decentralised MPC (as in any of the heuristically divided flexibility mechanisms) are independent, which implies that the nodes cannot support each other to reach the global optimum, resulting in a delay. This makes it challenging to use heuristically divided flexibility mechanisms in systems that require fast and efficient responses or constantly need to ensure that the global optimum is reached.

This confirms that the automatic flexibility mechanism designed by the authors in the form of Centralised MPC is a more efficient method to define joint constraints between several Nodes in a large-scale system than heuristic methods that still apply today.

To conclude this section, the following table has been created to summarise all the characteristics of the various algorithms that have been analysed in this document.

	Centralised MPC	**Decentralised MPC**
Convergence guarantee	Yes	Yes
Global optimality guarantee	Yes	No
Computational requirements	Medium (*)	Low
Execution time of one instant	Medium (*)	Low

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 October 2021 doi:10.20944/preprints202110.0457.v1

Response time (number of execution instants to reach the optimum)	Low	High(**)
Fulfilment of flexibility targets	High	Medium

Table 1: Comparison between Centralised MPC and Decentralised MPC

(*) It can be reduced with modern technologies.

(**) Only guaranteed if the global optimum can be reached or if it can be reached in an amount of time considered reasonable

7. Discussion

This research has offered a new method for automating the distribution of aggregate constraints between Nodes of a large-scale system. Although the use case studied is simple, with the main objective of allowing an efficient and straightforward analysis, the designed method can be extrapolated to other more complex large-scale systems without losing effectiveness in its mathematics.

It was demonstrated that the designed Centralised MPC, which has automatically distributed flexibility mechanisms, achieves optimal solutions, is convergent and produces more efficient results than other methods used in the control of large-scale systems with aggregate constraints between nodes. Effectiveness has been demonstrated by comparing one of the most historically widespread methods, Decentralised MPC, which prevents communication between Nodes and thus forces subdivisions of the flexibility mechanisms.

The flexibility mechanisms are optimally distributed across the instants of the prediction window in the MPC proposed by the authors (**Figure 8**). The performance achieved in this research presents promising results for industry sectors seeking to implement controllers in large-scale systems. One of the main characteristics of the proposed method, compared to traditional methods, is that it uses the advantages of today's technology without losing the benefits of Node modelling. In this way, it is possible to reach the desired results and distribute the constraints efficiently, without complex modelling or mathematics, achieving the most optimal solutions.

Author Contributions: Conceptualisation, A.J.d.R.; software, J.D.; validation, A.J.d.R. and J.D.; writing—original draft preparation, A.P.; writing—review and editing, A.P, A.J.d.R. and J.D.; supervision, A.J.d.R. and J.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: No data has been obtained from external sources.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ryan L Hartman. Flow chemistry remains an opportunity for chemists and chemical engineers. *Current Opinion in Chemical Engineering*. 2020, 29, 42-50; DOI: 10.1016/j.coche.2020.05.002.
2. Jämsä-Jounela, SL. Future Automation Systems in Context of Process Systems and Minerals Engineering. *IFAC-PapersOnLine*. 2019, 52, Issue 25, 403-408; DOI: 10.1016/j.ifacol.2019.12.570.
3. Fadaeenejad, M.; Saberian, A.M.; Fadaee, M.; Radzi, M.A.M.; Hizam, H.; AbKadir, M.Z.A. The present and future of smart power grid in developing countries. *Renewable and Sustainable Energy Reviews*. 2014, 29, 828-834; DOI: 10.1016/j.rser.2013.08.072.
4. Kaldellis, J.K.; Zafirakis, D. Optimum energy storage techniques for the improvement of renewable energy sources-based electricity generation economic efficiency. *Energy*. 2007, 32, Issue 12, 2295-2305. DOI: 10.1016/j.energy.2007.07.009. X
5. Arumugasamy, S.K.; Ahmad, Z. Model Predictive Control (MPC) and Its Current Issues in Chemical Engineering. *Chemical Engineering Communications*. 2012, 199, 472-511; DOI: 10.1080/00986445.2011.592446.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 October 2021 doi:10.20944/preprints202110.0457.v1

6. Brooks, K.; Westcott, M.; Bauer, M. A Combined MPC for Milling and Flotation – A Simulation Study. *IFAC-PapersOnLine*. 2019, 52, Issue 14, 24-29; DOI: 10.1016/j.ifacol.2019.09.158.
7. Garcia-Torres, F; Báez-Gonzalez, P.; Tobajas, J.; Vazquez, F.; Nieto, E. Cooperative Optimization of Networked Microgrids for Supporting Grid Flexibility Services Using Model Predictive Control- *IEEE Transactions on Smart Grid*. 2021, 12, no. 3, 1893-1903; DOI: 10.1109/TSG.2020.3043821.
8. Camacho, E.F.; Bordons-Alba, C. Model Predictive Control. *Springer-Verlag London*. 2007; DOI: 10.1007/978-0-85729-398-5.
9. Majanne, Y.; Model predictive pressure control of steam networks. *Control Engineering Practice*. 2005, 13, 1499-1505; DOI:10.1016/j.conengprac.2005.03.008.
10. Forbes, M.G.; Patwardhan, R.S.; Hamadah, H.; Gopaluni, R.B. Model Predictive Control in Industry: Challenges and Opportunities. *IFAC-PapersOnLine*. 2015,48, Issue 8, 531-538; DOI: 10.1016/j.ifacol.2015.09.022.
11. Jamshidi, M. Controls, Large-Scale Systems. *Encyclopedia of Physical Science and Technology (Third Edition)*. 2003, 675-686; DOI: 10.1016/B0-12-227410-5/00144-7.
12. Lauro, F; Moretti, F.; Capozzoli, A.; Panzieri. S. Model Predictive Control for Building Active Demand Response Systems. *Energy Procedia*. 2015, 83, 494-503; DOI: 10.1016/j.egypro.2015.12.169.
13. Hidalgo-Rodríguez, D. I.; Myrzik, J. Optimal Operation of Interconnected Home-Microgrids with Flexible Thermal Loads: A Comparison of Decentralized, Centralized, and Hierarchical-Distributed Model Predictive Control. *Power Systems Computation Conference*. 2018, 1-7; DOI: 10.23919/PSCC.2018.8442807.
14. Geidl, M.; Koeppel, G.; Favre-Perrod, P.; Klockl, B.; Andersson, G.; Frohlich, K. Energy hubs for the future. *IEEE Power and Energy Magazine*. 2007, 5(1), 24-30; DOI: 10.1109/MPAE.2007.264850.
15. Del Real, A. J.; Arce, A.; Bordons, C. An Integrated Framework for Distributed Model Predictive Control of Large-Scale Power Networks. *IEEE Transactions on Industrial Informatics*. 2014, 10(1), 197-209; DOI: 10.1109/TII.2013.2273877.
16. Lauro, F.; Longobardi, L.; Panzieri, S. An adaptive distributed predictive control strategy for temperature regulation in a multizone office building. *IEEE International Workshop on Intelligent Energy Systems (IWIES)*. 2014, 32-37; DOI: 10.1109/IWIES.2014.6957043.
17. Arnold, M.; Negenborn, R.R.; Andersson, G.; De Schutter, B. Distributed Predictive Control for Energy Hub Coordination in Coupled Electricity and Gas Networks. *Intelligent Infrastructures*. 2010, 42, 235-273; DOI: 10.1007/978-90-481-3598-1_10.
18. Trodden, P.; Richards, A. Cooperative distributed MPC of linear systems with coupled constraints. *Automatica*. 2013, 49, Issue 2, 479-487; DOI: 10.1016/j.automatica.2012.11.007.
19. Morosan, P-D.; Bourdais, R.; Dumur, D.; Buisson, J. Distributed MPC for Multi-zone Temperature Regulation with Coupled Constraints. *IFAC Proceedings Volumes*. 2011, 44, Issue 1, 1552-1557; DOI: 10.3182/20110828-6-IT-1002.00516.
20. Del Real, A. An integrated framework for distributed model predictive control of large scale networks. Applications to power networks. Doctoral dissertation, University of Seville: Spain. 2011.
21. Du, Y; Wu, J.; Li, S.; Long, C.; Paschalidis, I.C. Distributed MPC for Coordinated Energy Efficiency Utilization in Microgrid Systems. *IEEE Transactions on Smart Grid*. 2019, 10(2), 1781-1790. DOI: 10.1109/TSG.2017.2777975.
22. Ananduta, W.; Maestre, J.M.; Ocampo-Martinez, C.; Ishii, H. Resilient distributed model predictive control for energy management of interconnected microgrids. *Optimal Control Applications and Methods*. 2020, 41, 146– 169; DOI: 10.1002/oca.2534.
23. Bemporad A.; Barcelli D; Decentralized Model Predictive Control. *Networked Control Systems*. 2010, 406; DOI: 10.1007/978-0-85729-033-5_5
24. Wong, H. On the CMOS Device Downsizing, More Moore, More than Moore, and More-than-Moore for More Moore. *IEEE 32nd International Conference on Microelectronics (MIEL)*. 2021, 9-15; DOI: 10.1109/MIEL52794.2021.9569101.
25. Gommers, S.; Lazar, M. Smart decentralized MPC for temperature control in multi–zone buildings. *29th Mediterranean Conference on Control and Automation (MED)*. 2021, 415-420; DOI: 10.1109/MED51440.2021.9480233.
26. El Houda Hammami, D.; Maraoui, S.; Bouzrara, K. Implementation of MPC control for quadruple-tank system using LabView: A decentralized approach. *19th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*. 2019, 41-46; DOI: 10.1109/STA.2019.8717241.