# On the Use of Gradient-Based Solver and Deep Learning Approach in Hierarchical Control: Application to Grand Refrigerators

Xuan-Huy H Pham, François Bonne, Mazen Alamir

# Taylor & Francis Word Template for journal articles

Xuan-Huy PHAM[a,b]*, François Bonne[b] and Mazen Alamir[a*]

[a]*CNRS, Univ. Grenoble Alpes, Gipsa-lab, F-38000 Grenoble, France;*

[b]*Univ. Grenoble Alpes, IRIG-DSBT, F-38000 Grenoble, France;*

*Mazen Alamir: mazen.almir@grenoble-inp.com

*Xuan-Huy PHAM: pxh661995@gmail.com

# On the use of gradient-based solver and deep learning approach in hierarchical control: Application to grand refrigerators

This paper extends the work that has been recently studied on hierarchical control proposed by (Alamir Mazen 2017). This framework is designed to control interconnecting subsystems such as cryogenic processes or power plants. Based on the previous study, (Pham Xuan-Huy 2022) have shown that handling constraints and non-linearities could dispute the real-time feasibility of the approach. In order to reduce the computation time of the nonlinear model predictive controllers (NMPCs) of the local subsystems, two successful directions are investigated and combined, namely truncated fast gradient based NMPC approach and deep-neural-network-based approach. It is also shown that by doing so, the control updating period can be significantly reduced and the closed-loop performance is greatly improved. This paper can therefore be considered as a concrete implementation and validation of some key ideas in the design of real-time distributed NMPCs. All concepts are validated using the realistic and challenging example of a real cryogenic refrigerator.

Keywords: Hierarchical MPC, Fixed-point iteration, Real-time, Deep learning, Gradient-based solver, Cryogenic refrigerators.

## 1. Introduction

In nuclear fusion reactors or particle accelerators, cryogenic refrigerators play an essential role because of their ability to cool thermal loads on superconducting magnet-based components to maintain the functionality of the overall process. These installations are composed of several highly interacting subsystems that require truly effective control designs. In practice, there are many reasons why some changes (e.g., readjusting the local controllers, changing actuators, or even activating/inactivating a subsystem) need to be made at the subsystem level. These modifications usually come with disadvantages since any changes at the subsystem level are usually made without any proper assessment. Moreover, if the modification is not carefully done, The entire process might be unexpectedly destabilized. Thus, the need for a framework that provides the ability to

switch between different modes and facilitate modular modification is an emerging topic for large-scale system control.

With the purpose of having a framework suitable for cryogenic process specifications, (Alamir Mazen 2017) proposed a two-layer hierarchical structure, in which a network of coupled subsystems resides at the local layer (lower layer), and a coordinator resides at the coordination layer (upper layer). At the upper layer, the coordinator attempts to solve a central optimization problem with respect to the setpoint vector and sends its components to the corresponding subsystems. Essentially, the coordinator is assumed not to know any mathematical information about the subsystems (**modular privacy preservation requirement**), so the central cost evaluation process must be performed by fixed-point iteration that guarantees the consistency constraint on coupling signals. The results of (Pham Xuan-Huy 2022) shown that incorporating nonlinearities into local control problems can improve the performance of the overall framework. However, it can lead to losses of control performance if the computation becomes too complex and infeasible within the allowed computation time due to limited computational resources. This fact has been demonstrated by numerical simulations with the use of NMPC for the Brayton cycle. In fact, the method consisting in distributing the optimization over the real-life time is only valid if the computation time resulted from the distribution is compatible with the predefined updating periods $[k, k+1]\tau_u$, which must be long enough to cover the computation time.

The authors underlined that the computational burden in this framework is due to the resolution of local nonlinear MPC problems being repeated over the fixed-point iterations and for several set-points. This computational bottleneck is induced by using powerful but computationally expensive solvers such as Casadi (Diehl 2019) or Acado (Houska 2011). Based on this observation, this paper proposes two directions that could

be used if the computational time problem arises when implementing the proposed framework.

In order to replace such powerful but real-time incompatible optimization solvers, the simplest way is to use a sub-optimal solver. Indeed, according to (Richter 2011), a well-known gradient-based iterative solver is proposed to solve linear optimization control problem, by providing a technique to define lower iteration bound. This has prompted much works regarding the implementation aspect of MPC in embedded applications (Van Parys 2019, Kogel 2011, Van Parys 2019).

Another way to reduce the computation time is to approximate the control laws by piecewise affine functions (PWA) defined on a polyhedral partition of the feasible states (Mayne 2014, Seron 2000, Quevedo 2004). This method is also called explicit MPC. However, this property is only true if there is no nonlinearity present in the objective function or constraints. Besides, the complexity of the state space regions over which the control law is defined grows exponentially as the number of states increases, which makes this approach impractical for large-scale systems. Moreover, in the conventional explicit MPC methods, only the first action of the control sequence is approximated, whereas, in our proposed framework, the entire control sequence $\boldsymbol{u}_s$ is required for the fixed-point iteration.

Instead of approximating the nonlinear MPC by PWA functions, deep learning has become a popular choice due to its universal approximation property. Moreover, many works have demonstrated the effectiveness of these methods in many embedded applications (Bonzanini 2020, Chan 2021). Hence, the contribution of this paper is to address the computation time occurred when using NMPC in fixed-point iteration based hierarchical control. More precisely:

(1) First, a fast gradient-based algorithm is proposed. The performance of this solver is compared to available generic toolkits (such as Casadi/IPOPT ) in terms of optimization performance and computation time. This solver will be shown to be real-time compatible when used in the fixed-point-based hierarchical control framework.

(2) Then, deep learning approach is used to approximate the resulted control laws to further reduce the computation time of the most CPU-critical local controller.

(3) Finally, we will show that the reduction in computation time allows the control inputs to be updated more frequently, thus improving the closed-loop performance.

The paper is organized as follows: Section 2 describes the system and recalls the model predictive control as well as the proposed hierarchical control framework. Section 3 presents the truncated gradient-based optimization solver for solving MPC problem. Section 4 describes the deep learning approach, which is used to approximate the MPC laws. The simulation results and analysis are given in section 5 while section 6 concludes the paper and gives hints for future investigations.

## 2. Problem description

This section described the cryogenic system investigated in this paper. Fig. 1 shows the diagram of the station in the 400W configuration which can be decomposed into four components: Joule-Thomson cycle, Brayton cycle, pre-cooling Brayton cycle and warm compression station (WCS).

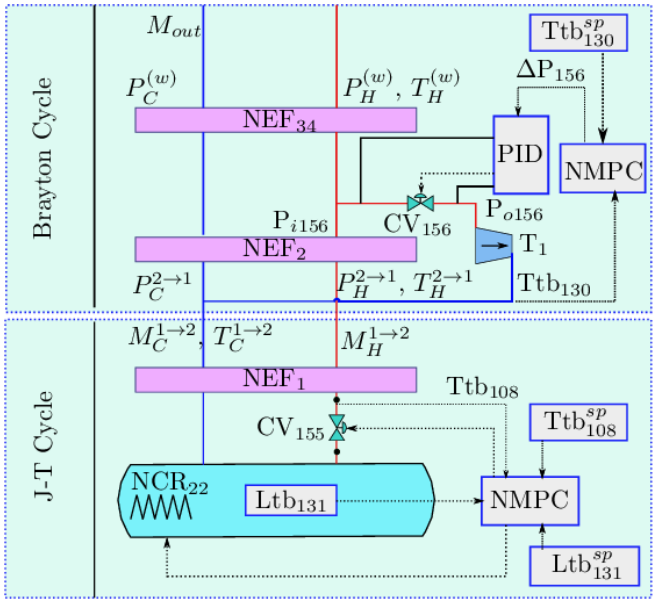


Figure 1: Diagram of the cold box of a cryogenic refrigerator.

Briefly speaking, the cryogenic refrigerator implements a closed thermodynamic cycle. The combination of the Brayton cycle and the J-T cycle is also called the cold box (Fig. 1). The gaseous helium flows clock-wise in two main pipelines which are hot pipe line (red line) and cold pipeline (blue line). The cooling power of the cryogenic refrigerator is generated by exchanging heat power in the fluid through a series of heat exchangers denoted by $\mathrm{NEF_x}$, and also by extracting thermal energy by using two turbines denoted by $\mathrm{T_1}$ (in Brayton cycle). The gaseous helium is partially liquefied after passing through the valve $\mathrm{CV_{155}}$ and rests in the helium bath, while the low temperature gaseous helium returns to the circuit. The main objective is to reject the disturbing heat power induced by the heat source denoted by $\mathrm{NCR_{22}}$. Concerning the simulation tools, the cryogenic toolbox *Simcryogenics* (F. Bonne 2020) developed for Matlab/Simulink environment is used to obtain the simulation result presented in this paper.

This system can be decomposed into an interconnecting network of four subsystems which is shown in Fig. 2. In this topology, there exists a set of subsystems indices denoted by $\mathcal{N} = \{1, \ldots, n_s = 4\}$ which is divided into two subsets $\mathcal{N}^{ctr}$ and

$\mathcal{N}^{unc}$. The indices belonging to the subset $\mathcal{N}^{ctr}$refers to the controlled subsystems, whereas the indices belonging to the subset $\mathcal{N}^{unc}$refer to the uncontrolled subsystems. In this decomposition, the controlled subsystems are: Joule-Thomson cycle, turbine $T_1$, while the remaining subsystems are uncontrolled. These subsystems are coupled through the coupling signals denoted by $v_{s'\to s}$ with $s' \in \mathcal{N}_s$ representing the set of subsystems' indices that affect the subsystem $S_s$.

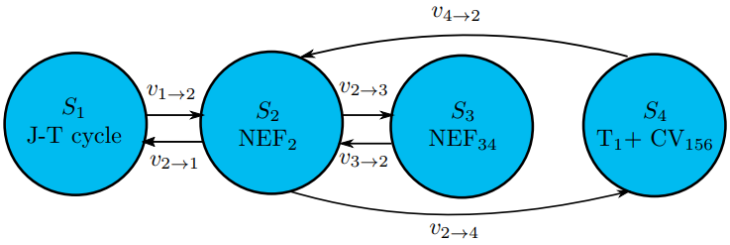


Figure 2: The interconnection between the subsystems of the cryogenic plant. The introduced set corresponding to this decomposition topology are $\mathcal{N} \coloneqq \{1, \dots, 4\}$; $\mathcal{N}^{ctr} \coloneqq \{1,4\}$; $\mathcal{N}^{unc} \coloneqq \{2,3\}$ ; $\mathcal{N}_1 \coloneqq \{2\}$; $\mathcal{N}_2 \coloneqq \{1,3,4\}$; $\mathcal{N}_3 \coloneqq \{2,4\}$; $\mathcal{N}_4 \coloneqq \{2,3\}$.

Before going any further, let us introduce the following notation which is extensively used in the sequel. Given a vector sequence $q_{i_1}, q_{i_2}, \dots, q_{i_{n\in Z}}$, the vector $q$ concatenating all these elementary vectors, provided by the corresponding concatenation operator col, is defined as follows:

$$q = \text{col}_{\,i\in\mathcal{I}} q_i \coloneqq \left[q_{i_1}^T, q_{i_2}^T, \dots, q_{i_n}^T\right]^T, \text{ with } \quad i_1 < i_2 < \cdots < i_n \in \mathcal{I} \tag{1}$$

Besides, the bold-faced notation represents the profile of a vector p over a prediction horizon of length N, namely:

$$\mathbf{p} = [p^T(k), \dots, p^T(k+N-1)]^T \in \mathbb{R}^{\mathrm{N}\cdot n_p} \tag{2}$$

with $n_p$ being the length of vector $p$.

The next subsections will described the controlled systems as well as the uncontrolled ones.

### *2.1. The controlled subsystems*

The first controlled subsystem in the studied system is the Joule-Thomson cycle. This subsystem is used to liquefy a portion of helium gas by passing the fluid through the valve $CV_{155}$. The dynamic model can be expressed by the nonlinear equation given below:

$$x_1^+ = f_1(x_1, u_1, w_1, v_1^{\text{in}}) \tag{3}$$

$$y_1 = h_1(x_1, u_1, v_1^{\text{in}}) \tag{4}$$

$$v_{1\to s'} = g_{1\to s'}(x_1, u_1, v_1^{\text{in}}) \tag{5}$$

where $x_s, u_1, w_1$ and $v_1^{\text{in}}$ are the deviations of state, control input, disturbance and the incoming coupling signal from the operation points $x_1^{\text{op}}, u_1^{\text{op}}, w_1^{\text{op}}$ and $v_1^{\text{in,opt}}$, respectively. The deviation of the incoming coupling signal $v_1^{\text{in}}$ is defined by $v_s^{\text{in}} = \text{col}_{s\in\mathcal{N}_1} v_{s'\to 1}$.

The detail of the control input, the regulated output and the disturbance are:

$$u_1 = \begin{bmatrix} \text{CV}_{156} \\ \text{NCR}_{22}^{(a)} \end{bmatrix} \qquad \text{y}_1 = \begin{bmatrix} \text{Ltb}_{131} \\ \text{Ttb}_{108} \end{bmatrix} \qquad w_1 = \text{NCR}_{22}^{(w)}$$

where $\text{Ltb}_{131}$(%) is the liquid helium level in the bath, $\text{Ttb}_{108}$ (K) is the temperature at the inlet of the J-T valve, $\text{CV}_{156} \in [-55,45]$ (%) is the opening position (expressed in terms of deviations) of the J-T valve, $\text{NCR}_{22}^{(a)} \in$ [0,100] (W) is the power of the heating actuator and $\text{NCR}_{22}^{(w)} \in$ [-100,100] (W)is the power of heating disturbance. Note that the heating power is divided in two variables, which are $\text{NCR}_{22}^{(\text{a})}$ and $\text{NCR}_{22}^{(w)}$. The first one is used as the control input, while the latter represents the disturbance power. Thus, the deviation of the heating power counted from its operation value $\text{NCR}_{22}^{\text{op}}$ can be expressed as:

$$\text{NCR}_{22} = \text{NCR}_{22}^{(a)} + \text{NCR}_{22}^{(w)} \tag{6}$$

The second controlled system is the turbine $T_1$. This component is a key element of the refrigerator since it allows the gas to expand by extracting its energy. The model of the turbine is given by the following static equation:

$$y_4 = h_4(\emptyset, u_4, v_4^{\text{in}}) \tag{7}$$

$$v_{4\to s'} = g_{4\to s'}(\emptyset, u_4, v_4^{\text{in}}) \tag{8}$$

The turbine's model does not employ involve any state vector, thus, the state vector is denoted by $x_4 = \emptyset$ in all the equations that are concerned. Also, $u_4 = \Delta\text{P}_{156}$ and $y_4 = \text{Ttb}_{130}$ are manipulated input and regulated output counted from its operation value $u_4^{\text{op}}$ and $y_4^{\text{op}}$, respectively. More precisely, $\text{Ttb}_{130}$ (K) is the temperature at the outlet of the turbine $T_1$, while $\Delta\text{P}_{156} \in$ [-1.5,12] (bar) is the pressure drop at the of the flow rate passing through the valve $\text{CV}_{156}$.

### *2.2.The uncontrolled subsystems*

The uncontrolled systems concerned in this plant are the heat exchangers $\text{NEF}_2(S_2)$ and $\text{NEF}_{34}(S_3)$, whose dynamics are given by:

$$x_s^+ = f_s(x_s, \emptyset, \emptyset, v_s^{\text{in}}) \tag{9}$$

$$y_s = h_s(x_s, \emptyset, v_s^{\text{in}}) \quad \text{for } s = 3 \tag{10}$$

$$v_{s\to s'} = g_s(x_s, \emptyset, v_s^{\text{in}}) \tag{11}$$

where $x_s$ and $v_s^{in}$ are the deviation of the states and incoming coupling signal from the operation points $x_s^{\text{op}}$ and $v_s^{\text{in,op}}$, respectively. Furthermore, the united incoming coupling vector $v_s^{\text{in}}$ that gathers all the coupling vectors coming from the neighbors that affect the dynamic of $S_s$ is defined by $v_s^{in} = \text{col}_{i\in\mathcal{N}_s} v_{s'\to s}$. Also, these subsystems do not have any control input or any disturbance input, their control input vectors and disturbance input are thus denoted by $u_s = \emptyset, w_s = \emptyset$ (for s $\in \{2,3\}$) in all the underlying equations. In

addition, the subsystem $S_3$ has one output which is the outgoing flow rate $y_3 + y_3^{\text{op}} = M_{out}$ (see Fig. 1), to be constrained to remain below 0.07 kg/s.

Next, the model predictive control, which is used to control the J-T cycle and the turbine is introduced.

### *2.3. Model predictive control*

This subsection recalls the MPC formulation that is used to control the controlled subsystem ($S_1$ and $S_4$).

Given an initial state vector $x_s(k)$, an incoming coupling profile $\boldsymbol{v}_s^{\text{in}}$, a disturbance profile $\boldsymbol{w}_s$ and any control profile $\boldsymbol{u}_s$ defined over the prediction horizon $[k, k+N]$, the corresponding nominal state trajectory is given by:

$$x_s(k+i+1) = f_s\big(x_s(k+i), u_s(k+i), v_s^{\text{in}}(k+i), w_s(k+i)\big) \quad \text{for } i = 0, \dots, N-1$$

The state profile $\boldsymbol{x}_s$ over a horizon of length $N$, can be defined by a straight-forward notation:

$$\boldsymbol{x}_s = \mathbf{f}_s(x_s(k), \boldsymbol{u}_s, \boldsymbol{v}_s^{\text{in}}, \boldsymbol{w}_s); \tag{12}$$

Similarly, the output profile and the outgoing coupling profile can be computed by using their corresponding equations associated to each subsystem (4), (5), (7) and (8) , namely:

$$\boldsymbol{y}_s = \mathbf{h}_s(\boldsymbol{x}_s, \boldsymbol{u}_s, \boldsymbol{v}_s^{\text{in}}); \tag{13}$$

$$\boldsymbol{v}_s^{\text{out}} = \mathbf{g}_s(\boldsymbol{x}_s, \boldsymbol{u}_s, \boldsymbol{v}_s^{\text{in}}); \tag{14}$$

At every instant $k$, the standard formulation of the optimal control problem are given by:

$$\mathcal{P}_s\colon \min_{\boldsymbol{u}_s} J_s^{\text{NMPC}}(x_s(k), \boldsymbol{u}_s, \boldsymbol{v}_s^{\text{in}}, \boldsymbol{w}_s, r_s)$$

$$= \min_{\boldsymbol{u}_s} \sum_{i=0}^{N-1} |y_s^{sp} - y_s(k+i)|_{Q_s}^2 + |u_s(k+i)|_{R_s}^2 \tag{15}$$

Subject:

$$\boldsymbol{x}_s = \mathbf{f}_s(\boldsymbol{x}_s(k), \boldsymbol{u}_s, \boldsymbol{v}_s^{\text{in}}, \boldsymbol{w}_s) \tag{16}$$

$$\boldsymbol{y}_s = \mathbf{h}_s(\boldsymbol{x}_s, \boldsymbol{u}_s, \boldsymbol{v}_s^{\text{in}}); \tag{17}$$

$$\boldsymbol{u}_s \in \mathbb{U}_s \tag{18}$$

where $Q_s \in \mathbb{R}^{n_y^{(s)} \times n_y^{(s)}}$ and $R_s \in \mathbb{R}^{n_u^{(s)} \times n_u^{(s)}}$ are weighting matrices on the output and control input.. The disturbance profile $\boldsymbol{w}_s$ (if any) has a constant value over the prediction horizon, which is equal to the current estimated disturbance value $w_s$.

This section has presented the nonlinear model predictive control, the next sub-section will recall the fixed-point-iteration based hierarchical control.

### *2.4. Recall on fixed-point-iteration based hierarchical control*

To begin, Fig. 3 describes the case of interest where a set of interacting subsystems indexed by $\mathcal{N} \coloneqq \{1, \dots, n_s\}$ is represented. Remember that this set is subdivided into two different subsets:

- A subset of controlled subsystems indexed by $\mathcal{N}^{ctr} \subset \mathcal{N}$ having each its control input vector and regulated output vector, denoted for any $s \in \mathcal{N}^{ctr}$ by $u_s$ and $y_s$ respectively.
- A potential complementary subset of subsystems that includes no control input denoted by $N^{unc} \coloneqq \mathcal{N} - \mathcal{N}^{ctr}$.

The dynamic of each subsystem $S_s$ is impacted through the so-called coupling signals $v_{s' \to s}$ coming from all exogenous subsystems belonging to the set $\{S_{s'}\}_{s' \in \mathcal{N}_s}$, with indices s' belonging to the set of indices $\mathcal{N}_s$ (set of indices of subsystems impacting $S_s$).

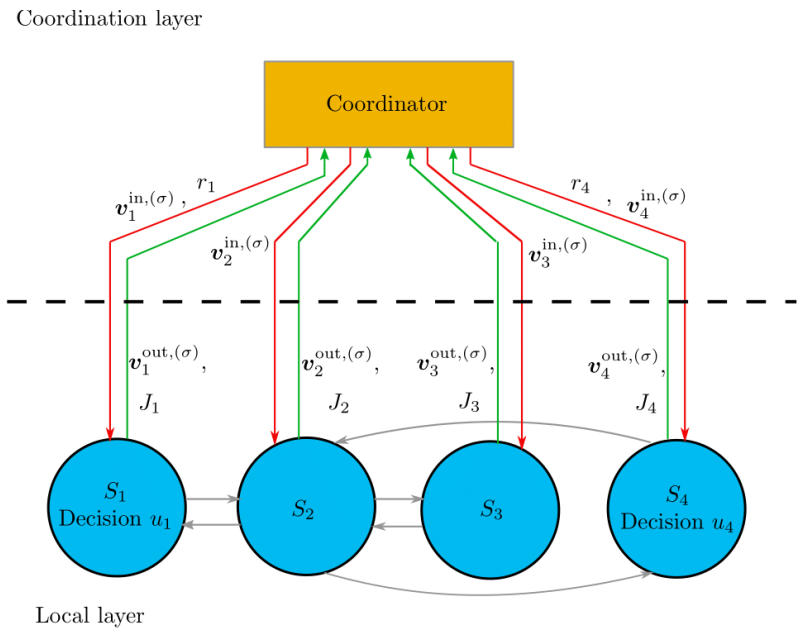


Figure 3: Synoptic view the hierarchical control architecture applied to the interconnecting network of the subsystems. Here, the 4-subsystem topology presented in Sect. 2 is used for the illustration of the general hierarchical framework.

Let $\boldsymbol{v}_s^{\text{in}}$ and $\boldsymbol{v}_s^{\text{out}}$ be the incoming/outgoing coupling profiles into and from the subsystem $S_s$ respectively. More precisely:

$$v_s^{\text{in}} := \text{col}_{s \in \mathcal{N}_s} v_{s' \to s}; \tag{19}$$

$$\boldsymbol{v}_s^{\text{out}} := \text{col}_{s' | s \in \mathcal{N}_{s'}} v_{s \to s'} \tag{20}$$

Note that the process described below is operated during each control updating period $[k, k+1]\tau_u$ (with $\tau_u$ being the sampling time constant), the state vectors $x_s$ and the disturbance vectors $w_s$ are thus dropped for a sake of brevity.

The generic formulation of the proposed framework has been well precisely defined in (Alamir Mazen 2017). However, it is essential to recall the overall hierarchical control framework. Let us begin with the following assumption:

---

ASSUMPTION 1: Each subsystem $S_s$, when given:

- a presumed incoming profile $\boldsymbol{v}_s^{\text{in}}$ and
- a given individual set-point $r_s$ (required if $s \in \mathcal{N}^{ctr}$),

---

can compute what would be:

- Its control profile $u_s$ (if it has) by solving a nonlinear optimization problem,
- Its resulting outgoing profile $\boldsymbol{v}_s^{\text{out}}$ and
- Its contribution $J_s$ to the central cost.

The central cost is assumed to be of the form:

$$J_c(r, v^{\text{in}}) \coloneqq \sum_{s \in \mathcal{N}^{ctr}} J_s(r_s, v_s^{\text{in}}) + \sum_{s \in \mathcal{N}^{unc}} J_s(v_s^{\text{in}}) \tag{21}$$

where $r \coloneqq \text{col}_{s \in \mathcal{N}^{ctr}} r_s$ and $\boldsymbol{v}^{\text{in}} \coloneqq \text{col}_{s \in \mathcal{N}} \boldsymbol{v}_s^{\text{in}}$. Note that typical regulation-based cost $J_s$ is defined for $s \in N^{ctr}$ while $J_s$ might represent a constraints violation indicator when $s \notin \mathcal{N}^{ctr}$.

More precisely, each time the coordinator sends $(r, \boldsymbol{v}^{\text{in}})$ to the subsystems, this allows the subsystems to compute (in parallel) their corresponding control profiles $\boldsymbol{u_s}$ (if $s \in \mathcal{N}^{\text{ctr}}$) and the outgoing coupling signal profiles $\boldsymbol{v}_s^{\text{out}}$ which is represented by the following form:

$$\boldsymbol{v}_s^{out} = \mathbf{g}_s^{out}(r, \boldsymbol{v}_s^{in}) \tag{22}$$

Recall that both bm $\boldsymbol{v}^{\text{in}}$ and $\boldsymbol{v}^{\text{out}}$ are composed of all the elementary profiles $\boldsymbol{v}_{s \to s'}$.

Hence, there is a matrix $G_{in}$ such that:

$$\boldsymbol{v}^{\text{in}} = G_{in} \cdot \boldsymbol{v}^{\text{out}} \tag{23}$$

By injecting (22) into (23), we obtain:

$$\boldsymbol{v}^{\text{in}} = G_{in} \cdot \mathbf{g}_{\text{out}}(r, \boldsymbol{v}^{\text{in}}(r)) \tag{24}$$

Consequently, the problem that needs to be solved exclusively by the coordinator can be stated as follows:

$$\text{r}^{\text{opt}} = \underset{r}{\text{argmin}}\, J_c(r, \boldsymbol{v}^{\text{in}}(r)) \tag{25}$$

$$\text{subject to: } \boldsymbol{v}^{\text{in}} = G_{in} \cdot \mathbf{g}_{\text{out}}(r, \boldsymbol{v}^{\text{in}}(r))$$

Since the coordinator does not have any mathematical knowledge of the subsystems the

fixed-point map represented by (24) cannot be analytically known to the coordinator neither is the expression of the cost function J. That is the reason why the enforcement of (24) for a given set-point $r$ is done through a round of iterations between the coordinator and the subsystems as initially suggested by (Alamir Mazen 2017). More precisely, a fixed-point-iteration-based algorithm is proposed to evaluate a central cost associated to a given set-point $r$. Briefly, the algorithm could be summarized as below:

(1) The coordinator starts by sending an initial guess $v_s^{\text{in},(\sigma=0)}$ regarding the incoming profiles,

(2) The subsystems compute their control profiles (if any) and the corresponding outgoing coupling profiles $\hat{v}_s^{\text{out},(\sigma)}$ as well as their local cost $J_s$,

(3) The subsystems send the outgoing coupling profiles $\hat{v}_s^{\text{out},(\sigma)}$ to the coordinator from which the coordinator can use to reconstruct the corresponding incoming coupling profiles $\hat{v}_s^{\text{in},(\sigma)}$ based on (23).

(4) To ensure the convergence of the iteration, a stabilizing filter or a residual-based iterative method is used to update the profile denoted by $v_s^{\text{in},(\sigma+1)}$. Reader is referred to the work of (Alamir Mazen 2017) for the synthesis of this filter.

(5) The iterations continue until one of the termination criteria $\epsilon := max\left(\left|v^{\text{in},(\sigma+1)} - v^{\text{in},(\sigma)}\right|\right) \leq \epsilon_{max}$ or $\sigma \geq \sigma_{max}$ satisfied.

Upon convergence of the above fixed-point iteration, the coordinator disposes of the value of the cost function for the current candidate of the set-point vector. Having the possibility to compute the cost associated to a given set-point, any derivative-free optimization algorithm can be used to solve the central problem (25) with respect to the decision variable $r$ (e.g. Genetic algorithm, BOBYQA, etc.).

## 3. Fast gradient method for solving NMPC problem

It has been shown that when a limited (computation time)/(hardware performance) is present, a truncated fast gradient might be beneficial to closed-loop performances. That is why this algorithm is briefly recalled here as it is in the heart of the forthcoming development.

Recall that each subsystem $S_s$, $s \in \mathcal{N}^{ctr}$ solves an optimization problem upon receiving a pair of $(r_s, v_s^{in})$ from the coordinator, combining with the estimated state $\hat{x_s}$ and the disturbance profile $w_s$:

$$\mathcal{P}_s: u_s^* = \underset{\{\boldsymbol{u_s} \in \mathcal{U}_s\}}{\operatorname{argmin}} J_s^{\mathrm{NMPC}}(\boldsymbol{u}_s, \xi_s) \tag{26}$$

where $J_s^{\mathrm{NMPC}}$ is the NMPC cost and $\mathcal{U}_s$ is the admissible set of control profiles $\boldsymbol{u}_s$. The vector $\xi_s$ encapsulates all parameters such as the estimated state $\hat{x_s}$, the set-point $r_s$, the incoming coupling profile $v_s^{in}$ and the disturbance profile $w_s$ (if any). These variables are considered frozen during the resolution of (15) – (17) and will be dropped in this section for a sake of compactness.

The implementation of the fast gradient method requires the gradient of the cost function at $J_s^{\mathrm{NMPC}}$ with respect to $\boldsymbol{u}_s$, which can be easily obtained by modeling the cost with CasADi and then computing its gradient $\nabla J_s^{\mathrm{NMPC}}$. The algorithm that is used to solve (15) – (17) is given by the following updating rule:

$$z_s^{i+1} = u_s^i - \gamma \cdot \nabla J_s^{\mathrm{NMPC}}\left(u_s^i\right) \tag{27}$$

$$u_s^{i+1} = \mathbf{Pr}\left(z_s^{i+1} + c \cdot \left(z_s^{i+1} - z_s^i\right), \mathcal{U}_s\right) \tag{28}$$

where $c \in (0,1)$ is the design variable and $\mathbf{Pr}(p, \mathcal{U}_s)$ is the projection of vector $p$ on the admissible set $\mathcal{U}_s$. The variable $\gamma$ is the step size that is calculated by using Barzilai-Borwein formula proposed below:

$$\gamma^{i+1} = \frac{\left|\left(\boldsymbol{u}_s^{i+1} - \boldsymbol{u}_s^{i}\right) \cdot \left(\nabla J_s^{\mathrm{NMPC}}\left(\boldsymbol{u}_s^{i+1}\right) - \nabla J_s^{\mathrm{NMPC}}\left(\boldsymbol{u}_s^{i}\right)\right)\right|}{\left|\nabla J_s^{\mathrm{NMPC}}\left(\boldsymbol{u}_s^{i+1}\right) - \nabla J_s^{\mathrm{NMPC}}\left(\boldsymbol{u}_s^{i}\right)\right|^2} \tag{29}$$

Also, the convergence of the algorithm could be improved when a restart mechanism is included. More precisely, the variable $u_s$ is restarted every $n_{rst}$ iteration, but it is noted that the frequency of restarts should depend on the cost function.

Finally, this method is summarized by Algorithm 1.

**Algorithm 1 : Fast conjugate gradient method**

**Input :** $i \leftarrow 0; c \in (0,1); \gamma^i \in (0,1); n_{rstr} \in N; u_s^i \leftarrow \mathbf{0}; z_s^i \leftarrow \mathbf{0};$

**For** $i \leftarrow 1, \dots, \mathrm{N_{max}}$ **do**

  $z_s^{i+1} = u_s^i - \gamma^i \cdot \nabla J_s^{loc}\left(u_s^i\right);$

  **If** $\mathrm{mod}(i, n_{cstr}) == 0$ **then**

    $u_s^{i+1} = \mathbf{Pr}\left(z_s^{i+1}, \mathcal{U}_s\right)$

  **else**

    $u_s^{i+1} = \mathbf{Pr}\left(z_s^{i+1} + c \cdot \left(z_s^{i+1} - z_s^i\right), \mathcal{U}_s\right)$

  **end**

  Compute $\gamma^{i+1}$ by (29)

**end**

The next section will present the Deep-learning approach, which can be used to approximate the model predive control laws to reduce the commutation time.

## 4. Neural-network-based NMPC

In this section, the objective is to derive a regression model that predicts the values of $u_s^*$ by basing on a learning data set in which the algorithm 1 is involved. The central idea here is to replace the implicitly defined control profile (15) – (17) by an explicit representation of the form $u_s^* = K_{NN}(\xi_s, \theta_s^*)$, where $\theta_s^*$ is the parameters that minimize the objective function given below:

$$\theta_s^* = \underset{\theta_s}{\mathrm{argmin}} \frac{1}{N_s^{data}} \sum_{i=1}^{N_s^{data}} \left|u_s^{*,(i)} - K_{NN}\left(\xi_s^{(i)}, \theta_s\right)\right|^2 \tag{30}$$

where $\left\{\left(\xi_s^{(1)}, u_s^{*,(1)}\right), \dots, \left(\xi_s^{(N_s)}, u_s^{*,(N_s)}\right)\right\}$ is the set of $N_s^{data}$ training data. In this section,

only one subsystem (the Joule-Thomson cycle) is considered, the subscript $s$ is thus omitted for the sake of simplicity. Once the network architecture is trained, the approximate DNN-based NMPC law $K_{NN}(\xi, \theta^*)$ can be used online to cheaply evaluate the optimal control input.

There are two common data-generation strategies, namely open-loop and closed-loop. In open-loop data generation, the set $\mathcal{E} \subset \{\mathcal{X} \times \mathcal{V}^{\text{in}} \times \mathcal{R} \times \mathcal{W}\}$ of possible states, incoming coupling profiles, disturbances and set-points could be created and the corresponding control profile $u$ computed that will be added together to establish a set of data $\mathcal{D} = \{\left(x^{(i)}, \boldsymbol{v}^{\text{in},(i)}, r^{(i)}, \boldsymbol{w}^{(i)}, \boldsymbol{u}^{(i)}\right)\}_{i=1}^{N}$. Although very simple, this strategy can result in non physically realistic instances being included in the training data. Closed-loop strategy, on the contrary, gathers data while running a closed-loop simulation under randomly drawn physically meaningful initial states. Indeed, the majority of large-scale cryogenic systems operate under a relatively small number of regimes or operating scenarios. Each operational scenario is characterized by a few regulated and/or constrained outputs and a few large magnitude disturbances that may frequently change, while the set-points are kept unchanged for a long period of time. Hence, we propose the following data generation procedure that performs off-line simulation using the system model under the control law to collect the operationally relevant training set $\mathcal{D}$:

(1) Randomly samples relevant values of $r$ and $w$ in their operational ranges.

(2) Run the closed-loop simulations with the above discussed hierarchical design at some chosen initial states with the created PRBS signals. The data is collected during the fixed-point iterations to capture the relationship between the control profile $u$ and the triplet $(r, x, v^{\text{in}})$.

The network is trained to minimize the mean squared error criteria below:

$$J_{NN}(\theta) = \frac{1}{2}\sum_{i=1}^{N_{tr}} |u^{(i)} - K_{NN}(\xi^{(i)}, \theta)|^2 \tag{31}$$

where $N_{tr} < N$ is the number of training observations. Indeed, Before the training process, the data set is passed through a series of data preparation techniques and finally separated into two subsets that contain $N_{tr}$ samples and $N_{val} = N - N_{tr}$ samples, which serve to train and validate the regression model. Recall that the vector $\xi^{(j)} \in \mathcal{E}$ encapsulates all the parameters $\boldsymbol{x}^{(i)}$, $\boldsymbol{v}^{\text{in},(i)}$, $\boldsymbol{r}^{(i)}$ and $\boldsymbol{w}^{(i)}$.

## 5. Numerical results

### *5.1.Comparison between truncated fast MPC and Casadi/IPOPT*

First, we compare the control performance given by the truncated fast gradient solver presented in Sect. 3 and IPOPT solver of Casadi. The 4-subsystem-decomposition described in Sect. 2 is reused to conduct the simulation presented in this section. In addition, the local controllers for the Joule-Thomson cycle ($S_1$) and the turbine $T_1$ ($S_4$) are nonlinear MPCs.

The performance of the Ipopt (CasADi) solver and the truncated gradient solver used to solve the local optimal control problems of $S_1$ and $S_4$ are compared together. This can be done by evaluating the open-loop performance indicated by $J_s^{\text{NMPC}}(u_s^*)$, where $u_s^*$ is the solution of (15) – (17). More precisely, the process below is realized:

(1) Create realistic set of state $x_s$, set-point $r_s$ and $v_s^{\text{in}}$ denoted by $\mathcal{D}^{\text{sle}} \coloneqq \{\left(x_s^{(i)}, r_s^{(i)}, \boldsymbol{v}_s^{(i)}\right)\}_{i=1}^{N_{dta}}$.

(2) Solve the problem (15) - (17) by using solver Ipopt and truncated gradient at triplets $\left(x_s^{(i)}, r_s^{(i)}, v_s^{(i)}\right)$ (for $i = 1, \dots, N_{dta}$).

(3) The open-loop performances $J_s^{\mathrm{NMPC},(i)}(u_s^{*,\mathrm{Ipopt}})$ and $J_s^{\mathrm{NMPC},(i)}(u_s^{*,\mathrm{grd}})$ of the solver Ipopt and truncated gradient are computed. Then, the average of performance ratio $\bar{J}$ between the two solvers is deduced, namely:

$$\bar{J} = \frac{1}{N_{dta}} \sum_{i=1}^{N_{dta}} \frac{J_s^{\mathrm{NMPC},(i)}\left(u_s^{*,\mathrm{solver}}\right)}{J_s^{\mathrm{NMPC},(i)}\left(u_s^{*,\mathrm{base}}\right)} \times 100\% \text{ with solver} \coloneqq \{\mathrm{grd},\mathrm{Ipopt}\} \tag{32}$$

The performance in terms of optimization and computation time is analyzed. Table 2 shows the maximum computation time $t_{\mathrm{cpt}}^{\max}$ and the open-loop performance $\bar{J}$ associated to several solver's configuration on tolerance error $\epsilon_{\mathrm{tol}}$ and maximum iteration $N_{max}$. The local costs $J_s^{\mathrm{NMPC},(i)}\left(u_s^{*,\mathrm{base}}\right)$ appearing at the denominator in (32) is chosen to be the ones associated to the IPOPT solver with the configuration of $\epsilon_{\mathrm{tol}} = 10^{-4}$ and $N_{\max} = 10$. The computation time for subsystem S1 when using the truncated gradient-based solver is significantly reduced, while the optimization performance are almost identical. This can be realized by the fact that the computation time given by the choice of $\epsilon = 10^{-4}$ and $N_{\max} = 10$ reduced from 4.76s to 0.0499s by using gradient-based solver with $N_{max} = 100$, whereas the performance index $\bar{J}$ is not too much changed.

Table 1: Comparison of the truncated gradient solver and the IPOPT solver.

| | NMPC of $S_1$ | | | | NMPC of $S_4$ | | | |
|---|---|---|---|---|---|---|---|---|
| Solver | $N_{\max}$ | $\epsilon_{\mathrm{tol}}$ | $\bar{J}$ | $t_{\mathrm{cpt}}^{\max}$[s] | $N_{\max}$ | $\epsilon_{\mathrm{tol}}$ | $\bar{J}$ | $t_{\mathrm{cpt}}^{\max}$[s] |
| Truncated gradient descend | 100 | _ | 100.2379 | 0.0499 | 100 | _ | 101.3418 | 0.011 |
| | 50 | _ | 100.2395 | 0.0398 | 50 | _ | 101.3419 | 0.008 |
| | 30 | _ | 100.2357 | 0.0322 | 30 | _ | 101.3418 | 0.0043 |
| | 10 | _ | 101.29 | 0.0246 | 10 | _ | 101.3418 | 0.0014 |
| Ipopt/Casadi | 5 | $10^{-1}$ | 99.999 | 2.746 | 5 | $10^{-1}$ | 100 | 0.0589 |
| | 10 | $10^{-1}$ | 100.002 | 3.756 | 10 | $10^{-1}$ | 100 | 0.0873 |
| | 10 | $10^{-4}$ | 100 | 4.76 | 10 | $10^{-4}$ | 100 | 0.1720 |

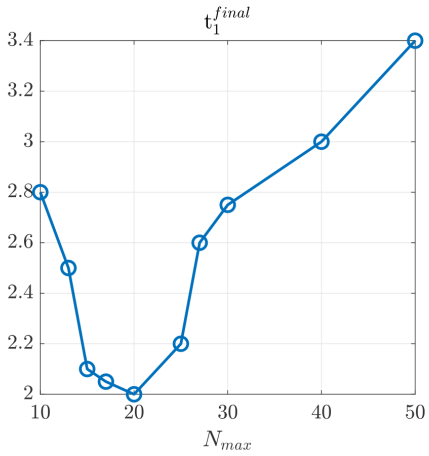


Figure 4: Evolution of computation time of $S_1$ needed for computing the optimal set-point $r^{\text{opt}}$ using the fast-gradient algorithm and the associated control profile $u_1$.

Furthermore, the real computation time with respect to the allowable maximum iteration $N_{\text{max}}$ when using fast-gradient based NMPC solver should be analysed. Fig. 4 shows the optimum computation time of $t_1^{\text{final}}$ for $S_1$ to compute the optimal setpoint $r^{\text{opt}}$ and the associated control profile $\boldsymbol{u_1}$ can be obtained by setting $N_{\text{max}} = 20$. The increase in computation time as $N_{\text{max}}$ decreases from 20 can be explained by the fact that the performance of the solver is significantly deteriorated, which prevents the convergence of the fixed-point iterations. For the simulation, $N_{\text{max}} = 30$ chosen for both NMPCs of $S_1$ and $S_4$.

### *5.2. Approximate NMPC by neural network*

In this subsection, the most time-consuming NMPC, which is the one of the J-T cycle ($S_1$), will be approximated by a deep neural network. The approach described in Sect. 4 is proceeded by beginning with the data preparation step. The data is collected during the closed-loop control simulation for the created profiles of $r_1$ and $w_1$, separately, and under the hierarchical control algorithm described in Sect. 2.4. Then, many deep neural network structures are used to approximate the control law based on the collected data.

After the data are gathered, the data pre-processing techniques are proceeded, such that: data balancing, data normalization, data Shuffling and data splitting. Once the data is ready, three feed-forward neural networks are trained. These configurations are set up so that each DNN has a different number of hidden layers, ranging from 1 to 3 hidden layers, with each layer having the same number of nodes, i.e., 25 nodes, denoted by NN-1-25, NN-2-25, and NN-3-25, respectively. The activation function at each node is the sigmoid function (other activation functions have been used but do not give any better performance). Concretely, each structure is trained for 10000 epochs with the prepared data set and is validated with the validation data set. The resilient back-propagation (RPROP) algorithm is used to train the neural network. Table 3 presents the learning performance for three DNN structures. The structure NN-2-25, which has the lowest mean squared error (MSE) is chosen to conduct the next simulation.

Table 2. The learning performance of several configuration of DNNs.

| Structure | NN architecture | MSE | Training time |
|---|---|---|---|
| NN-1-25 | [25 25 12] | 0.3192 | 2h47 |
| NN-2-25 | [25 25 25 12] | 0.2726 | 3h15 |
| NN-3-25 | [25 25 25 25 12] | 0.2996 | 3h50 |

### *5.3. Simulation result*

In order to facilitate the result interpretation, some performance indicators will be needed. First, the closed-loop performance indicator $J_c^{CL}$ is recalled, namely:

$$J_c^{CL} = \frac{1}{N_{sim}} \sum_{s\in\mathcal{N}} \sum_{i=1}^{N_{sim}} \left[ |y_s^{sim}(i) - r_s^d(i)|_{Q_c^{(s)}} + |u_s^{sim}(i)|_{R_c^{(s)}} + |\max\left(y_s^{sim}(i) - \overline{y}_s, 0\right)|_{Q_{cstr}^{(s)}} \right] (33)$$

where the weighting matrices $Q_c^{(s)}$, $R_c^{(s)}$ are chosen as followed:

- **Mode 1**: For disturbance rejecting scenario:

$$Q_c^{(1)} = \text{diag}(10^3, 10^3), R_c^{(1)} = \text{diag}(0,0), Q_{cstr}^{(1)} = 5 \cdot 10^9, Q_c^{(4)} = 10^3, R_c^{(1)} = 0$$

- **Mode 2**: For set-point tracking scenario:

$$Q_c^{(1)} = \text{diag}(10^6, 0.1), R_c^{(1)} = \text{diag}(0,0), Q_{cstr}^{(1)} = 5 \cdot 10^9, Q_c^{(4)} = 10^4, R_c^{(1)} = 0$$

The main advantages of the machine learning controller are in the implementation burden and computational efforts. Instead of solving the optimization problem several times in the fixed-point iterations, for several set-points to be evaluated and at each sampling instants, the NN-based controller only needs to evaluate the function $u_s^* = K\left(\xi\left(\mathrm{r}^{(i)}\right), \theta^*\right)$ at each iteration. Consequently, the computation time resulted by the implementation of NN-based controller is reduced. More precisely, the computation time imposed by the truncated gradient-based solver is reduced by factor 12 (the maximum computation times, needed to get the optimal set point, when using the gradient-based solver and the trained NN are 3.27 s and 0.27 s, respectively). In order to take advantage of this benefit, the control input can be updated more frequently, which will improve the control performance.

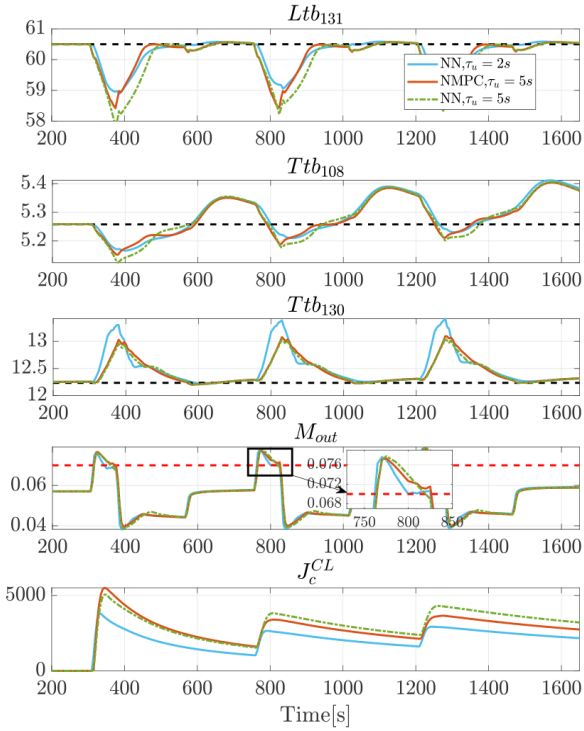

Figure 5: Output behaviors of the system in the case of disturbance rejecting under the coordination , in which NMPC and NN-based controller are implemented by $S_1$. The updating period is chosen to be $\tau_s = 5$ s and $\tau_s = 2$ s in order to compare the control performance.

Fig. 5 shows the output behaviors and the closed-loop control performance associated to the previous set-up of the local controllers, under the control updating period $\tau_u = 5$ s, and the one given by using the NN-based controller at $S_1$ under $\tau_u = 2$ s. In the comparison between the NN-based controller and the NMPC controller with the same updating period $\tau_u = 5$ s, the cumulative performance is dropped by 18 % (at time instant t= 1600 s of subfigure (5,1)). However, this performance is recovered and even improved approximately 50 % (at t = 1600 s) when the updating period is feasibly set to be at $\tau_u = 2$ s thanks to the use of NN-based controller. Finally, the use of NN-based controller is validated in the set-point tracking scenario illustrated in Fig. 6. It can be seen that the system behavior under the hierarchical control method with NN-based controller and with NMPC are similar. The NN-based controller can also mimic the behavior of the NMPC of $S_1$ in the set-point tracking case, which results the same system behaviors with less computation efforts (maximum computation time $t_{cmp}^{max}$ = 0.25 s).

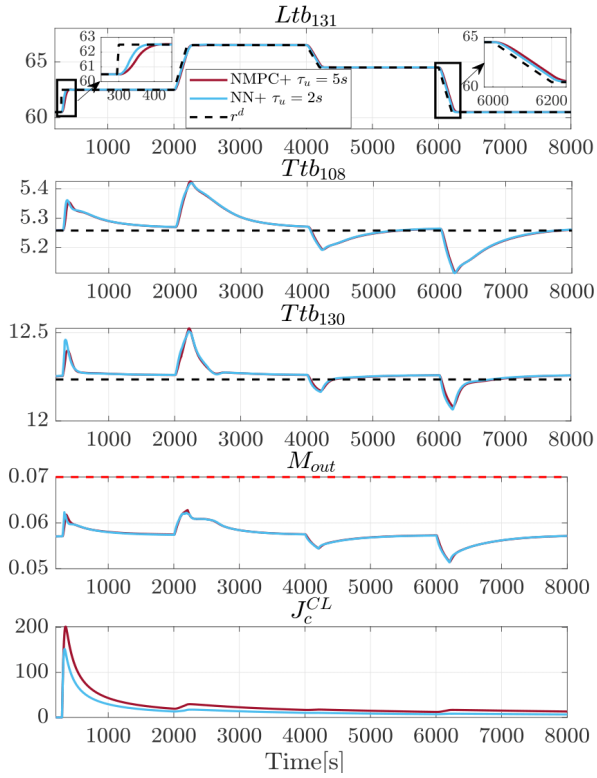


Figure 6: Output behaviors of the system in the case of set-point tracking under the coordination, in which NMPC and NN-based controller are implemented by $S_1$. The updating period is chosen to be $\tau_s = 5$ s and $\tau_s = 2$ s in order to compare the control performance.

## 6. Conclusion

In this paper, two methods have been proposed to reduce the computation time for solving the constrained nonlinear optimization problem at the local layer of the hierarchical control framework. The numerical results have demonstrated the effectiveness of the two approaches. More precisely, the computation time is reduced drastically by using the Truncated gradient method. Then, the control law is approximated by a deep neural network. Finally, the two approaches are then compared in terms of computation time and control performance, showing that the deep learning approach successfully approximates local control laws and allows for more frequent control

updates. On-going work aims to validate the control structure with a full cryogenic facility.

**References**


Alamir Mazen, Bonnay Patrick, Bonne François and Trinh Van-Vuong. "Fixed-point based hierarchical MPC control design for a cryogenic refrigerator." *Journal of Process Control* 58 (2017): 117-130.

Bonzanini, Angelo D and Paulson, Joel A and Graves, David B and Mesbah, Ali. "Toward safe dose delivery in plasma medicine using projected neural network-based fast approximate NMPC." *IFAC-PapersOnLine* 53, no. 2 (2020): 5279--5285.

Chan, Kimberly J and Paulson, Joel A and Mesbah, Ali. "Deep learning-based approximate nonlinear model predictive control with offset-free tracking for embedded applications." 2021. 3475--3481.

Diehl, J. Andersson and Joris Gillis and Greg Horn and J. Rawlings and M. "CasADi: a software framework for nonlinear optimization and optimal control." *Mathematical Programming Computation* 11 (2019): 1-36.

F. Bonne, S. Varin, A. Vassal, P. Bonnay, C. Hoa, F. Millet and J.-M , Poncet. "Simcryogenics: a Library to Simulate and Optimize Cryoplant and Cryodistribution Dynamics." *IOP Conference Series: Materials Science and Engineering.* 2020. 12-76.

Houska, Boris and Ferreau, Hans Joachim and Diehl, Moritz. "ACADO toolkit—An open-source framework for automatic control and dynamic optimization." *Optimal Control Applications and Methods* 32, no. 3 (2011): 298-312.

Kogel, Markus and Findeisen, Rolf. "A fast gradient method for embedded linear predictive control." *IFAC Proceedings Volumes* 44, no. 1 (2011): 1362-1367.

Mayne, David Q. "Model predictive control: Recent developments and future promise." *Automatica* 50, no. 12 (2014): 2967-2986.

Pham Xuan-Huy, Alamir Mazen, Bonne François and Bonnay Patrick. "Revisiting a fixed-point hierarchical control design for cryogenic refrigerators under constraints, nonlinearities and real-time considerations." *European Journal of Control* 63 (2022): 82-96.

Quevedo, Daniel E and Goodwin, Graham C and De Doná, José A. "Finite constraint set receding horizon quadratic control." *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal* 14, no. 4 (2004): 355-377.

Richter, Stefan and Jones, Colin Neil and Morari, Manfred. "Computational complexity certification for real-time MPC with input constraints based on the fast gradient method." *IEEE Transactions on Automatic Control* 57, no. 6 (2011): 1391-1403.

Seron, Maria M and De Dona, Jose A and Goodwin, Graham C. "Global analytical model predictive control with input constraints." *Proceedings of the 39th IEEE Conference on Decision and Control* . 2000. 154-159.

Van Parys, Ruben and Verbandt, Maarten and Swevers, Jan and Pipeleers, Goele. "Real-time proximal gradient method for embedded linear MPC." *Mechatronics* 59 (2019): 1-9.