Overlapping-Horizon MPC: A Novel Approach to Computational Constraints in Real-Time Predictive Control

Alberto Leva
DEIB, Politecnico di Milano, Italy

Simone Formentin
DEIB, Politecnico di Milano, Italy

Silvano Seva
DEIB, Politecnico di Milano, Italy

Abstract

Model predictive control (MPC) represents the state of the art technology for multivariable systems subject to hard signal constraints. Nonetheless, in many real-time applications MPC cannot be employed as the minimum acceptable sampling frequency is not compatible with the computational limits of the available hardware, *i.e.,* the optimisation task cannot be accomplished in one sampling period. In this paper we generalise the classical receding-horizon MPC *rationale* to the case where $n > 1$ sampling intervals are required to compute the control trajectory. We call our scheme *Overlapping-horizon MPC* – OH-MPC for short – and we numerically show its attitude at providing a tunable trade-off between optimisation quality and real-time capabilities.

2012 ACM Subject Classification Computer systems organization → Real-time systems; Information systems → Process control systems

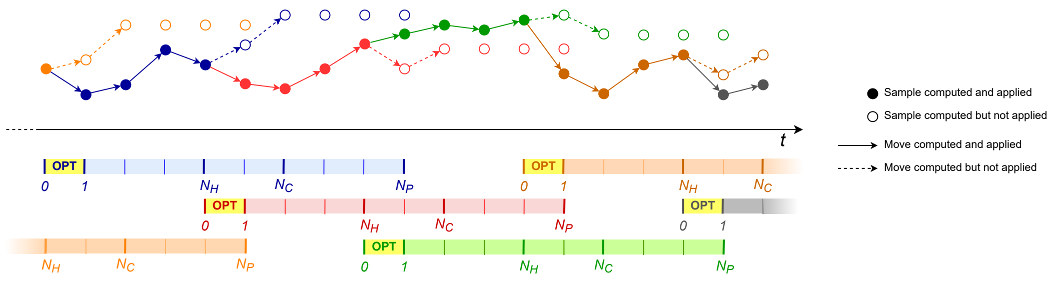
Keywords and phrases real-time control, model predictive control

Digital Object Identifier 10.4230/OASIcs.NG-RES.2022.3

1 Introduction and background

In modern control applications, Model Predictive Control (MPC) is the approach of election to optimal feedback regulation of multivariable (possibly nonlinear) systems subject to input and output constraints [1]. The key ingredient of all MPC strategies is the so-called *Receding-Horizon* policy, hereinafter RH-MPC for short. According to this policy, at each control step occurring at fixed period, the control signal trajectory is computed by considering its (predicted) impact on the state of the system over a future time window of finite length. Then, only the first sample of the optimal control sequence is applied to the system, whereas the whole input trajectory is re-optimised over a moved prediction horizon at the following sampling instant [9].

Despite the uncountable successful applications of such an approach [6], practical implementations of MPC schemes still present tough challenges for many real-world applications [5], among which: integration with data-driven model learning procedures [11], automatic tuning of MPC weights [8] and efficient on-line optimisation [12]. In particular, the latter represent a strong limitation for all applications where limited computational resources are available and a minimum sampling frequency is fixed by the dynamics at hand, in that the optimisation task *must* be accomplished in one period. Indeed, the expansion of MPC toward real-time systems on the one hand, and large-scale problems on the other, make the problem of allotting computational resource timely far more relevant than it was in the past, see e.g. [10].



t
OPT
0
1
N_H
N_C
N_P
● Sample computed and applied
○ Sample computed but not applied
Move computed and applied
Move computed but not applied

Figure 1 Role of the hold horizon N_H assuming that the optimisation problem (OPT) can be solved in one step; since in this particular example $N_H < N_C$, not all the computed control moves are applied; solid circles denote the control sequence actually fed to the process.

In [7], a suitable MPC scheme was proposed to overcome the above issues, by allowing the control action to be updated only "sporadically", standing on the assumption that an inner control layer is available to regulate the system, while the outer MPC is running in open loop. Another approach is to embed the predictive controller within a general event-based/asynchronous framework (see, e.g., [3] and [13]), thus limiting the number of computations but, at the same time, making the analysis unnecessarily complicated for a system whose signals are periodically sampled and with no network-related problems, like event-triggering or communication delays [14]. Moreover and most important, in sporadic and event-based strategies, when a new optimisation is required, this still must be done *within one control step.*

In order to deal with the problem of limited resources, in this work we propose a *generalisation* of the standard RH control scheme, for the case where $n > 1$ samples are required to perform the optimisation over the selected prediction horizon N_P. The proposed generalised scheme will be named *Overlapping-Horizon MPC* (OH-MPC) hereafter, in that its key idea is to start optimising the new input sequence while the previous control is applied – along this interval the old and new sequences are thus "overlapped", whence the name – and switch to the updated control trajectory as soon as this is available. In particular, by defining a *hold horizon* N_H as the number of samples of the computed input sequence that is actually applied to the system, we will discuss that, for $2n - 1 \leq N_H \leq N_P$, we can span with continuity (of course quantised in steps) between two *extrema*. One is the classical RH-MPC policy; the other (that we name herein *Open-Loop MPC* or OL-MPC for short) consists of applying the entire sequence of control samples as coming from the optimisation over the prediction window. In addition and most important, then, while spanning in between the said *extrema* we can always comply with the computational constraints. Finally, we will show that, if $n = 1$, our generalisation reduces to the traditional RH-MPC scheme, with all the related properties. In a nutshell, to summarise, we can outline our proposal by the following statements.

- Taking RH-MPC and OL-MPC as *extrema*, we introduce and exploit an additional degree of freedom in the form of applying only a part (N_H) of the computed control horizon.
- We build on this to propose a methodology for addressing the case in which the solution of the optimisation problem necessarily spans more than one control time step as dictated by the physics of the problem, which apparently makes RH-MPC infeasible.

The remainder of the paper is organised as follows. In Section 2, the notion of hold horizon N_H for a predictive control scheme is defined and the OH-MPC scheme is introduced and explained. Then, Section 3 discusses the presented ideas, also in a view to providing some

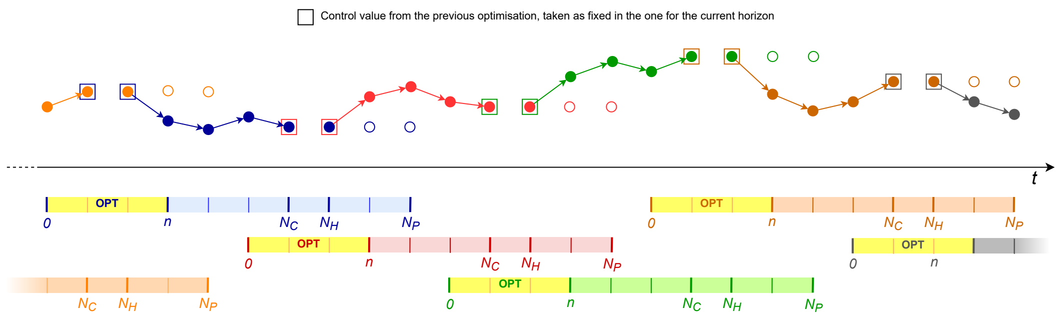

Control value from the previous optimisation, taken as fixed in the one for the current horizon
t
OPT
0
n
N_C
N_H
N_P

Figure 2 Horizons when the optimisation problem (OPT) cannot be solved in one step; here we set $N_C < N_H$ – hence all the computed control moves are applied – just for the sake of variety and completeness; again, solid circles denote the control sequence actually fed to the process.

application-oriented motivation for the additional degrees of freedom introduced. Section 4 provides a benchmark numerical example, to show that OH-MPC represents the best trade-off when limited computational resources are available. The paper is ended by some concluding remarks.

2 Overlapping-horizon MPC

In this section, we present the OH-MPC policy in general, and outline the corresponding algorithm. Let the system $\mathcal{S}$ to control be described in the discrete time domain by

$$x(t+1) = f(x(t), u(t)), \tag{1}$$

where $x \in \mathbb{R}^n$ represents the state vector, while f denotes a nonlinear function of the past state and input $u \in \mathbb{R}^m$. The standing assumption of the work is the following.

▶ **Assumption 1.** *The sampling time T_s of the application is dictated by the physical control problem, thus cannot be changed, and is so short that $n > 1$ samples are required to solve a state-feedback optimal control problem for* (1)*.*

In such a situation, not infrequent in the applications, the standard receding horizon policy cannot be applied. In this section, we will therefore derive our scheme as a generalisation of traditional MPC, to deal with this specific – yet potentially critical – case.

The bove said, in its most general form the OH-MPC problem is stated as

$$\begin{aligned}
&\min_{u(n),\ldots,u(N_c)} \quad \frac{1}{N_p - n} \sum_{h=n}^{N_p} \mathcal{L}(x(h), u(h)) \\
&\text{subject to :} \quad x(t+1) = f(x(t), u(t)),\ t = 1, \ldots, N_p, \\
&\qquad u(i) = \bar{u}(N_p - n + i),\ i = 1, \ldots, n, \\
&\qquad x(j) \in \mathcal{X},\ u(j) \in \mathcal{U},\ j = n, \ldots, N_p
\end{aligned} \tag{2}$$

As can be seen, the objective of the control strategy is to minimise a (possibly economic [4]) cost, expressed by the time average of a nonlinear function $\mathcal{L}(\cdot,\cdot)$ of states and inputs over a prediction horizon N_p, so that x and u are constrained to belong to some feasibility sets, called $\mathcal{X} \subset \mathbb{R}^n$ and $\mathcal{U} \subset \mathbb{R}^m$, respectively.

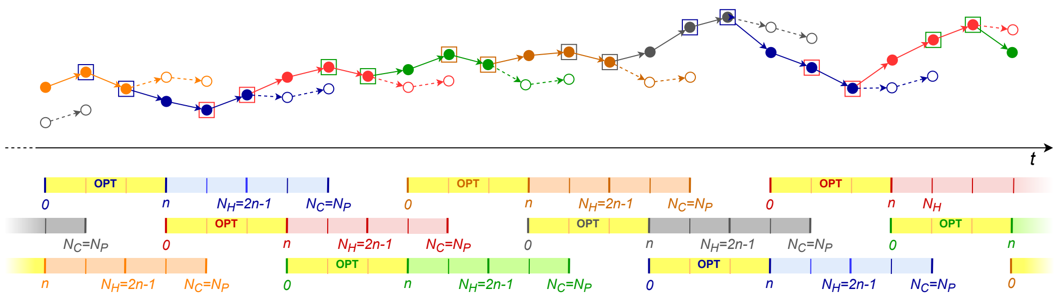


Figure 3 Horizons when the optimisation problem (OPT) cannot be solved in one step – particular but interesting case in which the optimisation problem can use all the available computational capability (we set here $N_C = N_P$ for simplicity, as this is a quite frequent choice); here too, solid circles denote the control sequence actually fed to the process.

The *key feature* of OH-MPC is that, since we assume n samples are required to return the optimisation results, the first $n-1$ samples of the input trajectory are set as the latest $n-1$ ones delivered at the previous iteration, namely, $\bar{u}(N_p - n + 1), \ldots, \bar{u}(N_p)$. Such a sequence is used within the optimisation routine to estimate (through the model f) the starting trajectory of the state x and then set the initial condition, i.e., the predicted value of $x(n)$, for the input sequence to be optimised, that is, $u(n), \ldots, \bar{u}(N_c)$, where N_c denotes the *control horizon*. We further assume that, when $N_c < N_p$, the sequence $u(N_c + 1), \ldots, u(N_p)$ is constant and equal to the most recent control $u(N_c)$.

Within this framework, there arises the need to introduce an additional degree of freedom N_H, called *hold horizon*, as the number of the computed control samples that are actually *applied* to the system. The role and meaning of the newly introduced hold horizon N_H is visually explained in Figure 1 for the traditional case of $n = 1$.

Even in this *scenario*, where the solution can be made available in one step, one might decide to apply a subset of the computed controls. For instance, in open-loop MPC (OL-MPC, hereafter), where the control input is applied in open-loop and updated only after the end of the prediction horizon, one might decide to apply only N_H samples of u (out of N_c) and then rerun the optimisation earlier. However, a typical choice of the hold horizon in OL-MPC with $N_C = N_P$ is $N_H = N_C$, namely, the input is optimised over the whole horizon and all the outcoming samples are applied to the system.

The choice of N_H becomes particularly interesting when $n > 1$ (see Figure 2). In fact, we can here highlight that the hold horizon must satisfy

$$2n - 1 \leq N_H \leq N_C. \tag{3}$$

The upper bound is encountered in those situations like OL-MPC where, no matter how large n is, the control system is run in open-loop (and, typically, $N_C = N_P$). The lower bound might become instead a rather restrictive constraint, in that it limits the minimum amount of input samples that have to be injected into the system in open-loop, due to the computational constraints. The limit case $N_H = 2n - 1$, where the control action is updated at the maximum possible frequency (dictated by n) is visually illustrated in Figure 3.

The overall strategy – that should now be clearly qualified as a generalisation of standard MPC, an aspect that will be further discussed in the next section – can be summarised as per Algorithm 1.

Algorithm 1 The OH-MPC algorithm.

```
   /* Problem acquisition & setup                                    */
 1 acquire the guaranteed No. $n$ of steps to optimize;
 2 acquire the horizons $N_P$, $N_C$, $N_H$;
 3 acquire the model of the problem, i.e., $\mathcal{L}(\cdot,\cdot)$, $f(\cdot,\cdot)$;
 4 acquire the initial conditions $x(0)$, $\bar{u}(t),\ t = N_P - n + 1, \dots, N_P$;
   /* Execution                                                      */
 5 while control system is running do
 6     solve (2) and collect $u(n), \dots, u(N_c)$;
 7     set $u(i) = u(N_C),\ i = N_C + 1, \dots, N_P$;
 8     compose the sequence
           $\mathrm{u}(t) = \begin{cases} \bar{u}(N_P - n + t), & t = 1, \dots, n, \\ u(t), & t = n, \dots, N_H \end{cases}$
 9     apply the composed sequence to the system $\mathcal{S}$ in (1);
10     when the sequence ends, thus the new optimisation time is reached, set $x(N_H)$ as
         the new $x(0)$ and $\bar{u}(t) = \mathrm{u}(t),\ t = 1, \dots, N_P$;
11 end
```

3 Discussion and motivation

We devote this section to briefly discuss the possibilities opened by the OH-MPC policy, also providing – compatibly with the proposal-oriented scope of this paper – some operational motivation for its adoption.

To this end, we first observe that by suitably setting the involved horizons, OH-MPC specialises to both known problems and new ones, of importance discussed below, to exploit the introduced additional degrees of freedom. In particular, if $n = 1$ is feasible, the following staements hold true.

- With $n = 1$ and $N_H = 1$, OH-MPC apparently reduces to the classical RH-MPC problem.
- With $n = 1$ and $N_H = N_P$, the so-called "open-loop MPC" (OL-MPC) is obtained; here, we call this "one-step-compute" OL-MPC to stress the condition on n.
- In the latter case, taking N_P as given, choosing $N_C < N_P$ is the one degree of freedom to reduce the size of the optimisation problem.

This said, let us briefly review alternatives to the proposed OH-MPC in its application case of election, i.e., when $n = 1$ is not feasible, and for reasons too long to discuss herein, OL-MPC is not considered reliable enough. These alternatives are substantially two. One can either downsize the optimisation problem by reducing the prediction and/or the control horizon, or replace the said problem with local equivalents (for example, linearising in the vicinity of conveniently chosen operating points) that require less effort to be solved.

More interesting, and motivating for the presented research, is therefore to study the case in which $n = 1$ is infeasible, but at the same time – once again for a variety of possible reasons that we are not treating in this paper – the optimisation problem must be solved *as is*. Here, OH-MPC can be fruitfully exploited to provide the needed additional degrees of freedom.

With reference to Figure 2, we assume for the scope of this work that optimisations occur only when some horizon elapses, i.e., we exclude for the moment techniques involving event-triggered optimisations like the sporadic one above.

This said, first the hold horizon N_H can be made both larger and smaller than the control one N_C. This allows to use the former to dictate the (constant) cadence of optimisations, while - for the latter - one can take the value dictated by the optimisation problem definition.

Second, when an optimisation is in progress, the control samples applied to the process come from the previous one, which – as will be shown – is tendentiously better than e.g. just holding the control signal till a new sequence is ready.

Third, once n is reliably obtained e.g. by profiling techniques aimed at WCET (Worst Case Execution Time) estimation [2], N_H allows to optimally use all the available resources. In detail, setting $N_P = N_C$ and $N_H = 2n - 1$ allocates all the computation time to perform the maximum number of optimisations – as illustrated in Figure 3 – compatibly with the time needed to compute one.

4 A proof-of-concept example

We now show a simulation case study – deliberately minimalistic – to witness the usefulness of OH-MPC. The system to be controlled is the linear time-invariant plant

$$\begin{cases} x_1(t+1) &=& 0.9 \cdot x_1(t) + 0.1 \cdot u(t) \\ x_2(t+1) &=& 0.6 \cdot x_1(t) + 0.4 \cdot x_2(t) \\ y(t) &=& x_2(t), \end{cases} \tag{4}$$

while the control problem we wish to solve is of the form in (2), with

$$\mathcal{L}(x(t), u(t)) = x(t)^T Q x(t) + u(t)^T R u(t),$$

$R = 1$, and

$$Q = \begin{bmatrix} 10 & 0 \\ 0 & 100 \end{bmatrix}.$$

In the addressed problem, we also set a constraint on the value of the control variable u, that must lie within the [-1,1] range. No constraints on x are given, namely $\mathcal{X} \equiv \mathbb{R}^2$. We consider the achieved closed-loop properties in terms of *disturbance rejection* using different model predictive controllers, via an experiment with a unitary matched load disturbance applied to the process at $t = 0$ and removed at $t = 30$. The prediction horizon N_P is set equal to 9 steps.

Case study no. 1. Let us assume that the computation time needed for the solution of the control problem amounts to three steps, thus $n = 3$.

In order to fairly assess the performance of OH-MPC, we consider a comparison among the following strategies, all with $N_P = N_C$:

1. an *oracle (in fact infeasible) solution*, given by an **OL-MPC** approach, where we assume – contrary to the OH-MPC hypothesis – that the solution can be computed in one step, that is $n = 1$, and $N_H = N_P = N_C$;

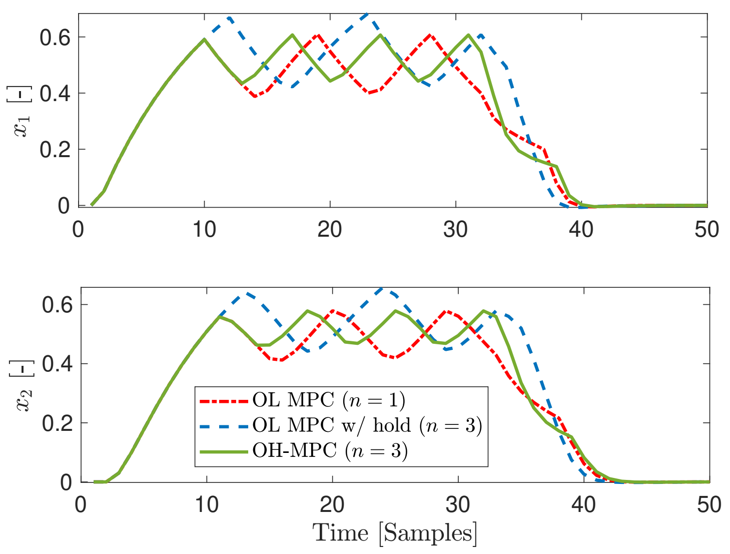


Figure 4 *Case study no. 1.* Time responses of state x to a step disturbance with different MPC strategies: OL-MPC (unfeasible), OL-MPC with final n-step holding and OH-MPC (feasible). OH-MPC clearly provides the best feasible performance.

2. a *baseline feasible solution*, in which the constraint $n = 3$ is taken into account, named **OL-MPC w/ hold**. This solution is obtained starting from the one above, but, at the beginning of a new optimization, the control variable is held equal to the latest available control sample for $n - 1$ steps, waiting for the new solution;
3. the **OH-MPC** approach with $N_H = N_P = N_C$.

The time histories of the state trajectories are illustrated in Figure 4, where it can be clearly seen that the OH-MPC solution remains limited as compared to the baseline one, even if it is not able to reach the attenuation level provided by OL-MPC. It is however worth stressing that this OL-MPC is calculating the optimal solution in one step only, which is infeasible under the assumption that $n = 3$.

The fact that OH-MPC can be considered as a good trade-off solution (namely, the best alternative if the constraint $n = 3$ is active) is further confirmed by comparing the value of the optimal cost

$$J = \sum_{t=1}^{N} x(t)^T Q x(t) + u(t)^T R u(t),$$

where N denotes the length of the whole experiment, in Figure 5.

Case study no. 2. Considering the same system of the previous example, we now suppose that the designer's desire is to use all the available computational power to run the controller at the maximum frequency, however under the physical constraint that the computation time needed for the solution of the control problem amounts to five steps ($n = 5$), which can be considered even more challenging than the previous situation, if $N_P = 9$. Again, we consider a comparison with the *baseline* strategy OL-MPC w/hold, but now – to use all the available time – $N_H = 2n - 1 = 9$.

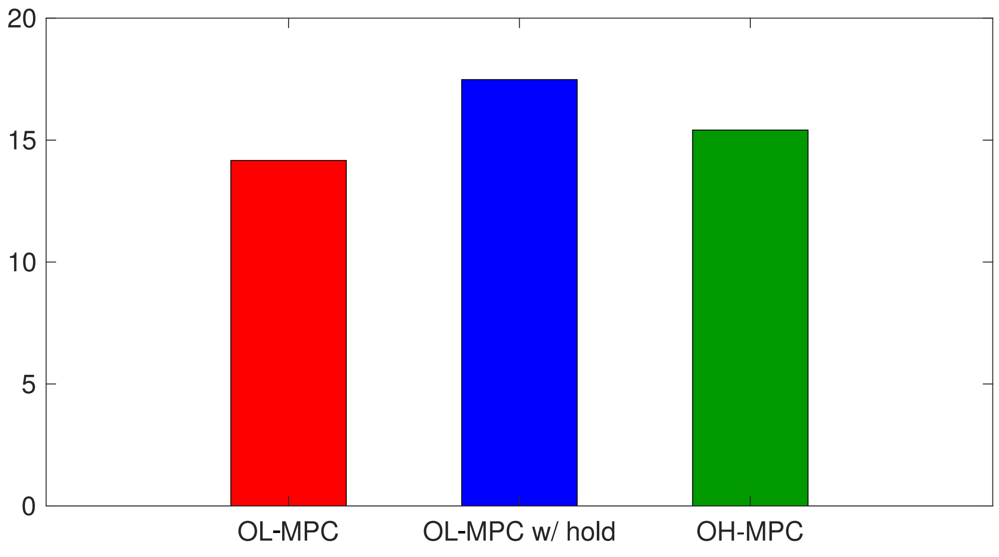


Figure 5 *Case study no. 1.* Cost J for different MPC strategies: OL-MPC (unfeasible), OL-MPC with final $(n-1)$-step holding and OH-MPC (feasible). OH-MPC clearly provides the best feasible performance.

The time histories of the state trajectories are illustrated in Figure 6, where it can be observed that the OH-MPC solution remains better than the baseline also in such a critical scenario. A confirmation of this fact can be found in Figure 7, where the cost J is highlighted, thus confirming OH-MPC gets closer to the ideal situation.

Summing up, OH-MPC appears to yield intermediate results between the (infeasible) one-step-computing solution ($n = 1$) and the baseline one based just on holding the last control value.

5 Conclusions and future work

We presented an MPC scheme, named *Overlapping-Horizon* MPC to comply with the case in which the optimisation problem cannot be solved in one control time step, or said otherwise, its solution can be guaranteed to terminate only within a number $n > 1$ of such steps. The addressed case is potentially critical and occurs whenever computational resource limitations can be relevant, whence the usefulness of OH-MPC in real-time control systems.

Although the presented research is still at a preliminary stage, OH-MPC definitely exhibits interesting properties, in particular not requiring to modify the optimisation problem with respect to its “original” formulation, nor to alter the sampling time with respect to the value dictated by the control system physics (another desirable propoerty in the real-time case).

Future work will be directed toward a formal analysis of the OH-MPC scheme, possibly articulating the study per characteristics of the controlled system and/or cost function, as well as toward an engineered realisation, suitable for implementation and testing on real plants.

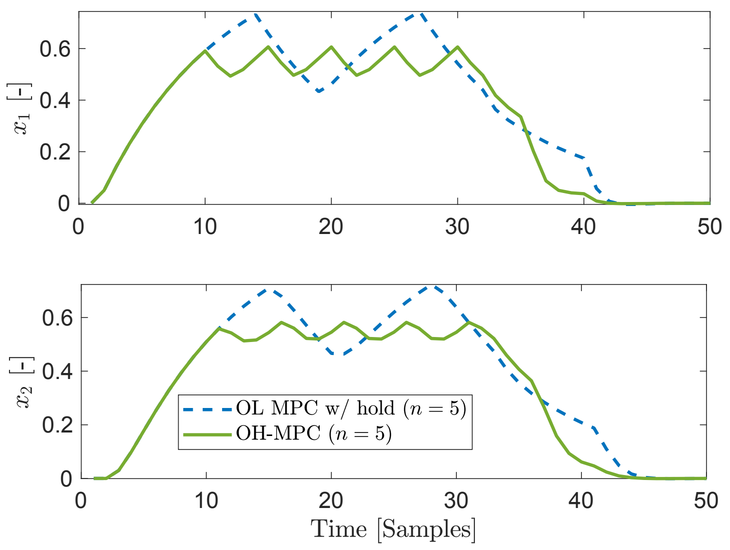


Figure 6 *Case study no. 2.* Time responses of state x to a step disturbance with different MPC strategies: OL-MPC with final n-step holding and OH-MPC.

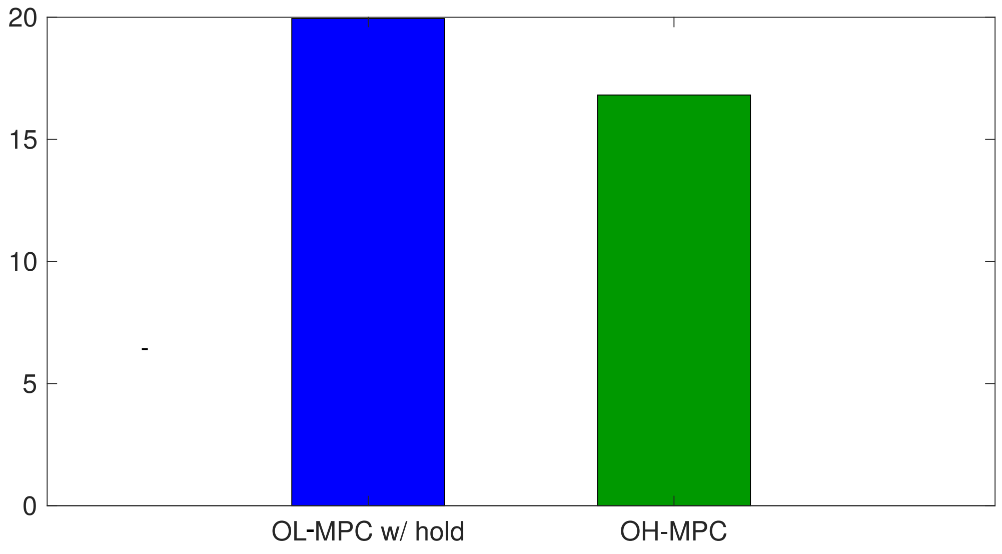


Figure 7 *Case study no. 2.* Cost J for different MPC strategies: OL-MPC with final $(n-1)$-step holding and OH-MPC (feasible).

References

1 F. Allgöwer and A. Zheng. *Nonlinear model predictive control*. Birkhäuser, 2012.

2 A. Bonci, S. Longhi, G. Nabissi, and G.A. Scala. Execution time of optimal controls in hard real time, a minimal execution time solution for nonlinear SDRE. *IEEE Access*, 8:158008–158025, 2020.

3 F.D. Brunner, W. Heemels, and F. Allgöwer. Robust event-triggered MPC for constrained linear discrete-time systems with guaranteed average sampling rate. *IFAC-PapersOnLine*, 48(23):117–122, 2015.

4 M. Ellis, H. Durand, and P.D. Christofides. A tutorial review of economic model predictive control methods. *Journal of Process Control*, 24(8):1156–1178, 2014.

5 M.G. Forbes, R.S. Patwardhan, H. Hamadah, and R.B. Gopaluni. Model predictive control in industry: Challenges and opportunities. *IFAC-PapersOnLine*, 48(8):531–538, 2015.

6 S.K. Lahiri. *Multivariable predictive control: Applications in industry*. John Wiley & Sons, 2017.

7 A. Leva, F.M. Benzi, V. Magagnotti, and G. Vismara. Sporadic Model Predictive Control. *IFAC-PapersOnLine*, 50(1):4887–4892, 2017.

8 A. Lucchini, S. Formentin, M. Corno, D. Piga, and S.M. Savaresi. Torque vectoring for high-performance electric vehicles: an efficient MPC calibration. *IEEE Control Systems Letters*, 4(3):725–730, 2020.

9 J. Mattingley, Y. Wang, and S. Boyd. Receding horizon control. *IEEE Control Systems Magazine*, 31(3):52–65, 2011.

10 P. Pazzaglia, C. Mandrioli, M. Maggio, and A. Cervin. DMAC: Deadline-miss-aware control. In *Proc. 31st Euromicro Conference on Real-Time Systems*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

11 D. Piga, M. Forgione, S. Formentin, and A. Bemporad. Performance-oriented model learning for data-driven MPC design. *IEEE control systems letters*, 3(3):577–582, 2019.

12 S.J. Wright. Efficient convex optimization for linear MPC. In *Handbook of model predictive control*, pages 287–303. Springer, 2019.

13 J. Yoo and K.H. Johansson. Event-triggered model predictive control with a statistical learning. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2019.

14 D. Zhang, P. Shi, Q. Wang, and L. Yu. Analysis and synthesis of networked control systems: A survey of recent advances and challenges. *ISA transactions*, 66:376–392, 2017.