Control Period Adaptation for Resource-Constrained MPC Applications

Marcello Domenighini[1,2], Paolo Pazzaglia[1], Christoph Mark[1],
Kevin Schmidt[1], Laura Beermann[1], Alessandro Vittorio Papadopoulos[2]

***Abstract*— The integration of control applications into cloud and edge expands the capabilities of modern control systems, but also introduces variability in shared resource availability and competition with other applications, posing new challenges for control design. This paper presents a multi-mode Model Predictive Control (MPC) framework tailored for resource-aware systems. By treating the controller period as a scalable parameter, our approach dynamically adjusts control accuracy and computational complexity in response to changing resource and state-space conditions. Unlike existing event- and self-triggered strategies, our multi-mode design allows users to actively manage trade-offs between computational load and control quality. We provide feasibility and stability guarantees for the proposed control framework and demonstrate its effectiveness in a simulated cart-pole system, showcasing significant improvements in computational resource efficiency without compromising control performance.**

I. Introduction

Advancements in computing and communication technologies have significantly influenced the design and implementation of modern control systems. Traditionally, control applications relied on specific, dedicated onboard hardware for real-time control tasks. Now, the growing accessibility of cloud and edge computing allows for complex control algorithms to be deployed remotely, reducing the need for powerful local hardware and the associated costs [1], [2], [3], [4]. This shift toward distributed control architectures enables more advanced control strategies, but also creates unique challenges in managing the shared resources for computation, communication and memory access.

In a distributed control setting, the variability in resource availability is a major concern. Multiple concurrent applications compete for limited resources, leading to fluctuations in computational capacity and communication bandwidth, ultimately affecting the responsiveness of the control task. This advocates for a shift from traditional "static" control paradigms toward resource-aware designs that adapt to the current resource availability while maintaining system stability and performance.

Related Work: Previous research has tackled resource-constrained control through various triggering strategies. Thanks to its flexibility, Model Predictive Control (MPC) is often chosen as underlying control strategy. In event-triggered control schemes [5], the controller is activated when specific state-space conditions are met. While effective on average, this approach can lead to intermittent peaks of execution, which may be undesirable in real-time applications where predictability is essential or when budgets are not flexible. Alternative event-driven "rollout" MPC approaches exist [6], [7] where the transmission schedule is optimized to produce sparse or sporadically changing input signals. In [8], the rollout idea is combined with a "token bucket" model for communication resources. In [9], the period is selected between multiple controllers running in parallel. Nonetheless, the focus remains on saving communication resources, and the approaches are still computationally intensive.

With self-triggered approaches, the controller determines its next activation time based on current conditions [10], [11], balancing resource usage and performance. Such adaptive rate is nonetheless coupled with the underlying optimization problem, which makes it difficult to control the activation rates independently. Varying the prediction horizon has been explored with a similar goal, e.g. in [12], [13]. However, this requires recompiling the control function at runtime, with possibly significant impact on the execution time.

Estimating the execution time of an MPC is difficult. Mathematical methods exist for relevant solvers, see e.g. [14], [15], but most control-theoretical results rely on a zero-time assumption. In [16], an input buffer is introduced, which covers possible delays in communication or computation. In [17], the size of the input buffer is automatically adjusted based on the deviation between the predicted and actual state-space conditions; in [18], the same is done based on the effect of disturbances. In a recent work [19], the execution time is taken into account by shifting the application of the optimal input by one step. As in [10], the time step is variable and an allocation mechanism ensures that it matches the time slot that is actually available.

Differently from existing self-triggered approaches, in this paper we couple the sampling frequency of the controller with a multi-mode design. A simple multi-mode MPC design is found in [20], where the controller alternates between remote and local MPC implementations with different sampling rates to address communication disruptions. In their setup, however, the local controller only serves as a safety fallback, leaving limited control over the computational trade-off under variable resource constraints.

Contribution: We present a resource-aware MPC strategy with a variable sampling period across multiple modes, enabling the controller to balance accuracy and computational load in response to variable state and resource conditions. Theoretical guarantees of feasibility and stability

[1] Robert Bosch GmbH, Germany
[2] Mälardalen University, Västerås, Sweden

Acknowledgements: This work was supported by the ITEA4 project 22013 OpenSCALING (grant #01IS23062A), by the Swedish Research Council (VR) with the PSI project No. #2020-05094, and the Knowledge Foundation (KKS) with the MARC project No. #20240011.

are provided under arbitrary mode switching patterns. The proposed solution allows flexibility in managing the trade-off between control quality and computational load. We illustrate in simulation the practical advantages of the proposed framework, showing significant improvements in resource efficiency while maintaining high control performance.

Notation: In context of MPC, x_k refers to the plant state at step k, whereas $x_{i|k}$ denotes its i-step ahead prediction. Optimal quantities resulting from an optimization problem are indicated with a $*$, e.g., $x^*_{i|k}$. For some integers $a, b \in \mathbb{N}$ with $b > a$ we define the index set $\mathbb{I}_a^b = \{a, \dots, b\}$. A sequence of state predictions is defined as $\mathbf{x}_k = \{x_{i|k}\}_{i \in \mathbb{I}_0^N}$. Given a square matrix $P \in \mathbb{R}^{n \times n}$, the expression $P \succ (\succeq)\, 0$ identifies P as a positive (semi-)definite matrix. For a vector $x \in \mathbb{R}^n$ we define the weighted two-norm as $\|x\|_Q^2 = x^\top Q x$.

II. Problem Formulation

We consider a linear time-invariant dynamics governed by

$$\dot{x}(t) = A_c\, x(t) + B_c\, u(t), \quad \forall t \in \mathbb{R}_{\geq 0}, \tag{1}$$

where $x(t) \in \mathbb{R}^{n_x}$ is the system state, $u(t) \in \mathbb{R}^{n_u}$ the control input, A_c and B_c the corresponding dynamic matrices.

State measurements are provided by a sensor with period $h > 0$, i.e., $x_k = x(t_k)$ is produced at time $t_k = kh$, $\forall k \in \mathbb{N}_{\geq 0}$. For compactness, we will also use integer indices $k \in \mathbb{N}_{\geq 0}$ to denote the discrete time instants t_k.

An actuator updates the control input applied to the plant periodically, with the same period h as the sensor, such that $u(t) = u_k$ for $t \in [t_k, t_{k+1})$, $\forall k \in \mathbb{N}_{\geq 0}$. The discrete-time description of (1) at the sampling instants then is

$$x_{k+1} = A\, x_k + B\, u_k, \tag{2}$$

where the system matrices are obtained as

$$\begin{bmatrix} A & B \\ 0 & I \end{bmatrix} = \exp\left(\begin{bmatrix} A_c & B_c \\ 0 & 0 \end{bmatrix} h \right). \tag{3}$$

Assumption 1: There exists a controller $u_k = K\, x_k$, such that system (2) is asymptotically stable, i.e., for given matrices $Q \succ 0$ and $R \succ 0$

$$\exists P \succ 0 :\ A_K^\top P A_K + Q_K - P \preceq 0, \tag{4}$$

with $A_K = A + BK$ and $Q_K = Q + K^\top R K$.

The control inputs u_k are the result of our proposed *multi-mode* MPC, consisting of M different *modes* (i.e., implementations) of MPC, where each mode is indexed in the set $\mathcal{M} = \{1, \dots, M\}$. The control design and the switching mechanism between modes are analyzed in detail later in Section III, but for the purposes of the problem formulation, some features are anticipated here.

First, each MPC mode $\mu \in \mathcal{M}$ is executed with period $h_\mu = \sigma_\mu h$, with $\sigma_\mu \in \mathbb{N}_{>0}$. The deadline for the successful completion of its execution is set equal to the period h_μ. The modes have different periods; within each period, only one mode can be active. Mode switches are only possible at the end of the period. All modes share the same prediction horizon N, and the time step size for the prediction is equal to h. Finally, differently from classic MPC implementations, when an iteration of the control mode completes its execution, the entire sequence of predicted control inputs is provided to the actuators. Such values are stored in a local memory, and the actuator applies them sequentially until they are overwritten by a subsequent control execution.

A. Resource Model

The controller runs on a server or platform, shared with other applications. Computational resources, e.g., a percentage of the cores utilization, must be divided among the concurrent applications. The successful execution of an iteration of the MPC controller requires to perform a certain amount of computations; to complete them in time, a minimum budget of computational resources must be provided, spread over the time interval when it is executing.

The required amount of resources may be variable, especially during the transient of the controlled system. We assume that, for each mode $\mu \in \mathcal{M}$, a corresponding worst-case amount $\beta_\mu \in \mathbb{R}_{>0}$ of computational resources must be always provided, to ensure a successful completion within the deadline. Also, due to the shared setting of the computational platform hosting the control functionality, the resources available for the controller may change over time.

At any time, the platform orchestrator (i.e., the entity managing the assignment of the resources to applications) assigns a resource budget $b(t) \geq 0$ to the control application. The allocated budget is modeled as the sum of two contributions

$$b(t) = \phi(t) + \delta(t), \tag{5}$$

where, $\phi(t) \geq 0$ is the resource budget readily available at time t, while $\delta(t)$ is a variable quota that models an additional degree of freedom on the part of the application. With $\delta(t) > 0$, we model an additional budget actively requested on top of $\phi(t)$, while $\delta(t) < 0$ represents an amount actively released from $\phi(t)$, because not necessary. When starting a new iteration at time k with mode μ, the allocated resource budget must satisfy the constraint

$$\int_{t_k}^{t_k + h_\mu} b(t)\, \mathrm{d}t \geq \beta_\mu, \tag{6}$$

ensuring completion of every execution within its deadline.

The free budget $\phi(t)$ depends on the variable load of the other applications hosted on the platform and is thus incontrollable by the MPC application and treated as a disturbance. To exclude pathological cases, we assume that the variability of the resources is slower than the sampling rate of the controller, and free resources cannot drop abruptly. The additional budget $\delta(t)$ is instead a variable that the control application can directly influence. When $\delta(t)$ is positive, it may, e.g., trigger the action of ramping up an additional core or server that is left idle otherwise. Fig. 1 shows an example of resource usage, while the control application changes the operating mode from λ to μ.

For some systems $\delta(t) \leq 0$, i.e., only freeing resources is possible. In this case, $\phi(t)$ represents an upper bound to the resource budget available for computation, effectively constraining via (6) which modes can be currently active.

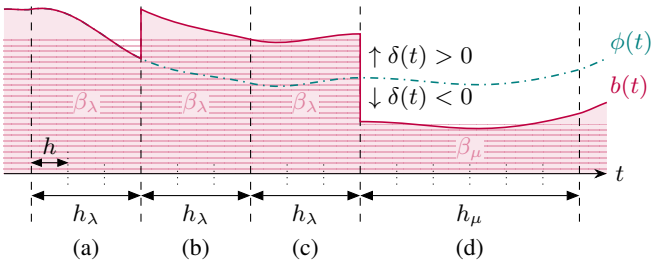

Fig. 1. Resource budget allocation. Within each control period, the allocated budget (shaded areas) must be greater or equal than the minimum budget β_μ (dashed areas) required by the selected control mode. If the free budget is insufficient, additional resources are requested (b, c). If the free budget is high enough, extra resources can be actively released (d).

B. Requirements

The MPC modes must be designed and chosen to guarantee stability in every condition. Moreover, the budget constraint in (6) must be satisfied at each iteration, so that the control application can run properly. Requesting (resp. releasing) resources is associated with a cost (resp. reward). Here, we consider a simple proportional cost

$$c(t) = W_r\, \delta(t). \tag{7}$$

Modes with higher frequency generally provide better control performance but require a higher budget (and cost) to run. Modes (and thus budgets) must be selected to maximize control performance and minimize cost, within the limits allowed by the stability requirement. Providing a detailed trade-off solution to this problem is outside the scope of the paper, but some guidelines are discussed in Section III-C.

III. Control Design

A. Multi-Mode MPC Design

The proposed multi-mode MPC design accounts for the actual resource budget, by creating modes with different periods and providing a seamless transition between such modes. In this section, the MPC formulation of a generic mode and its behavior during mode switching are presented.

Consider an arbitrary sampling instant k, where a new execution of control mode $\mu \in \mathcal{M}$ is triggered. By design, k is also the deadline of the previous instance, where mode $\lambda \in \mathcal{M}$ is running (μ and λ might be the same mode).

Unlike standard MPC implementations, the iteration of the chosen mode μ executes over σ_μ prediction time steps h before the new iteration is called. For each sampling instant in the continuous-time interval $[t_k, t_k + h_\mu)$—while the current iteration is executing—the actuator applies to the plant the corresponding elements of $\{u^*_{i|k-\sigma_\lambda}\}_{i=\sigma_\lambda}^{\sigma_\lambda+\sigma_\mu-1}$, taken from the optimal input sequence $\mathbf{u}^*_{k-\sigma_\lambda}$ that was computed at the previous iteration. At time t_k+h_μ, when the new optimal sequence $\mathbf{u}^*_k = \{u^*_{i|k}\}_{i=0}^{N-1}$ is made available, the value at $i = \sigma_\mu$ is applied, and then the successive ones, iteratively until the deadline of the next iteration. A graphical interpretation of this pattern is provided in Fig. 2.

The optimization problem for the MPC mode μ at time k is formally detailed as follows. Given the state vector x_k and the optimal sequence $\mathbf{u}^*_{k-\sigma_\lambda}$ obtained at previous iteration,

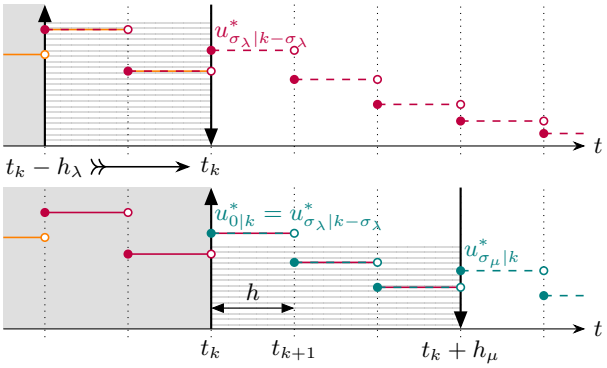

Fig. 2. Transition at t_k from mode λ (above) to mode μ (below). Dashed areas represent the current execution, while shaded areas the past execution. During each controller's period, the optimal input values from the previous iteration are applied while the control function is executed. These values are enforced during the first steps of the prediction.

for a prediction horizon N we aim to find the new optimal state and input sequences $\mathbf{x}^*_k \in \mathbb{R}^{n_u \times N+1}$ and $\mathbf{u}^*_k \in \mathbb{R}^{n_u \times N}$ minimizing the cost function

$$V(\mathbf{x}_k, \mathbf{u}_k) = \sum_{i=0}^{N-1} l(x_{i|k}, u_{i|k}) + V_f(x_{N|k}), \tag{8}$$

with stage cost $l(x,u) = \|x\|^2_Q + \|u\|^2_R$ and terminal cost $V_f(x) = \|x\|^2_P$, with P, Q, and R as in Assumption 1. The MPC problem reads

$$\min_{\mathbf{u}_k,\, \mathbf{x}_k} \quad V(\mathbf{x}_k, \mathbf{u}_k) \tag{9a}$$

$$\text{s.t.} \quad x_{i+1|k} = A\, x_{i|k} + B\, u_{i|k}, \qquad \forall i \in \mathbb{I}_0^{N-1} \tag{9b}$$

$$u_{i|k} = u^*_{\sigma_\lambda+i|k-\sigma_\lambda}, \qquad \forall i \in \mathbb{I}_0^{\sigma_\mu-1} \tag{9c}$$

$$x_{i|k} \in \mathbb{X}, \qquad \forall i \in \mathbb{I}_0^{N-1} \tag{9d}$$

$$u_{i|k} \in \mathbb{U}, \qquad \forall i \in \mathbb{I}_0^{N-1} \tag{9e}$$

$$x_{N|k} \in \mathbb{X}_f, \tag{9f}$$

$$x_{0|k} = x_k. \tag{9g}$$

where $\mathbb{X} \subset \mathbb{R}^{n_x}$ and $\mathbb{U} \subset \mathbb{R}^{n_u}$ are polytopic state and input constraints containing the origin, and $\mathbb{X}_f$ is such that the following assumption holds.

Assumption 2: The terminal set $\mathbb{X}_f$ is positively invariant for dynamics (2) under the terminal controller $u = K\, x$, i.e.:

$$A_K\, x \in \mathbb{X}_f, \quad \forall x \in \mathbb{X}_f, \tag{10}$$

and satisfies $\mathbb{X}_f \subseteq \mathbb{X}$, $K\, x \in \mathbb{U}$ for all $x \in \mathbb{X}_f$.

The formulation in (9c) takes into account the fact that, while executing the current iteration for σ_μ steps, the previous control inputs are used by the actuator. The new optimal control input sequence is thus designed such that $u^*_{i|k} = u^*_{\sigma_\lambda+i|k-\sigma_\lambda}$, for $i \in \mathbb{I}_0^{\sigma_\mu-1}$. After completion at time $k+\sigma_\mu$, the optimal control inputs sequence $u^*_{i|k}$, $i \in \mathbb{I}_{\sigma_\mu}^{N-1}$ is updated at the actuator level. At startup, a constant input value $u_k = u_{\text{init}}$ shall be applied for $k \in \mathbb{I}_0^{\sigma_\mu-1}$. To avoid pathological cases where not enough stored control inputs are available until the current execution is completed, we provide the following assumption.

Algorithm 1 Control loop

Initialization

$(\mathrm{m}, \mathrm{p}) \leftarrow \mu_{init} \in \mathcal{M}$ ▷ Initialize current and prev. modes
$(\sigma_\mathrm{m}, \sigma_\mathrm{p}) \leftarrow \sigma_{\mu_{init}}$ ▷ Initialize corresponding mode periods
$\mathbf{u}^*_{init} \leftarrow \{u_{init}\}_{i=0}^{N-1}$ ▷ Initialize control inputs
$i \leftarrow 0$ ▷ Initialize input counter
$k \leftarrow 0$ ▷ Initialize time step counter

Main control loop ▷ Running at sampling rate h
while True **do**
  if $i == \sigma_\mathrm{m}$ **then** ▷ At the deadline instant
    Store$(\mathbf{u}^*_{k-\sigma_\mathrm{m}})$ ▷ Fill memory with results of (9)
    $(\mathrm{p}, \sigma_\mathrm{p}) \leftarrow (\mathrm{m}, \sigma_\mathrm{m})$ ▷ Update values for previous mode
    Update$(\mathrm{m}, \sigma_\mathrm{m})$ ▷ Check if current mode is updated
    $i \leftarrow 0$ ▷ Reset input counter
  end if
  if $i == 0$ **then** ▷ At activation instant
    Run (9) ▷ Call problem (9) with proper inputs
  end if
  $u_k \leftarrow u^*_{i+\sigma_\mathrm{p}|k-i-\sigma_\mathrm{p}}$ ▷ Apply stored control input
  $i \leftarrow i + 1$
  $k \leftarrow k + 1$
end while

Assumption 3: For each pair $(\mu, \lambda) \in \mathcal{M}$, $\sigma_\mu + \sigma_\lambda \leq N$.

The overall control routine, including switching between modes, is detailed as pseudocode in Algorithm 1.

B. Formal Guarantees

So far, we introduced three modifications to the standard MPC design: the delayed application of the optimal inputs after the deadline, the extended controller period, and the possible transition to a new mode. These aspects do not alter the main ingredients of standard MPC design. The basic results of the recursive feasibility and asymptotic stability are preserved. The following assumption however is necessary.

Assumption 4: The MPC optimization problem (9) admits a feasible solution at time step $k = 0$ with $x_{0|0} = x_0$.

Theorem 1 (Feasibility): Consider system (2) under control input $u_{k+i} = u^*_{i|k}$, $i \in \mathbb{I}_0^{\sigma_\mu}$ from (9). If Assumptions 1–4 are satisfied, then the MPC optimization problem (9) is recursively feasible for all triggering times $k \geq 0$.

Proof: We prove the claim by induction. We start from the feasibility at time $k = 0$ (Assumption 4), then we show that if problem (9) is feasible at a generic triggering time k, it will be feasible at the next triggering instant as well.

Let σ_μ be the period of the control execution triggered at k, and $k + \sigma_\mu$ be the next trigger instant. At time $k + \sigma_\mu$, a feasible shifted candidate solution for mode μ is

$$\mathbf{u}_{k+\sigma_\mu} = \{u^*_{\sigma_\mu|k}, \dots, u^*_{N-1|k}, \; K\,x^*_{N|k}, KA_K\,x^*_{N|k}, \dots, KA_K^{\sigma_\mu-1}\,x^*_{N|k}\} \quad (11)$$

where the last σ_μ elements are appended terminal controllers. Applying (11) to (9b) results in the shifted state sequence

$$\mathbf{x}_{k+\sigma_\mu} = \{x^*_{\sigma_\mu|k}, \dots, x^*_{N-1|k}, \; x^*_{N|k}, A_K\,x^*_{N|k}, \dots, A_K^{\sigma_\mu}\,x^*_{N|k}\}. \quad (12)$$

In view of feasibility at time k, constraints (9d) are verified for all $\{x_{i|k+\sigma_\mu}\}_{i=0}^{N-\sigma_\mu-1}$. Due to the terminal constraint (9f), it holds that $x_{N-\sigma_\mu|k+\sigma_\mu} = x^*_{N|k} \in \mathbb{X}_f \subseteq \mathbb{X}$ which, together with the invariance property of Assumption 2, verifies (9d) for all $\{x_{i|k+\sigma_\mu}\}_{i=N-\sigma_\mu}^{N-1}$ and (9f) for $x_{N|k+\sigma_\mu}$.

Analogously, the input constraints (9e) are verified for all $\{u_{i|k+\sigma_\mu}\}_{i=0}^{N-\sigma_\mu-1}$. For $i \in \mathbb{I}_{N-\sigma_\mu}^{N-1}$, we apply the terminal controllers $K\,x_{i|k+\sigma_\mu}$. From the shifted candidate solution we have that $x_{N-\sigma_\mu|k+\sigma_\mu} \in \mathbb{X}_f$, and by invariance of $\mathbb{X}_f$, $\{x_{i|k+\sigma_\mu}\}_{i=N-\sigma_\mu+1}^{N-1} \in \mathbb{X}_f$ as well. From Assumption 2 we have that $K\,x \in \mathbb{U}$ for all $x \in \mathbb{X}_f$, therefore the input constraints (9e) are verified for all $\{u_{i|k+\sigma_\mu}\}_{i=N-\sigma_\mu}^{N-1}$. ■

Theorem 2 (Asymptotic Stability): If Assumptions 1–4 hold, then the origin of system (2) under control input $u_{k+i} = u^*_{i|k}$, $i \in \mathbb{I}_0^{\sigma_\mu}$ from (9) is asymptotically stable.

Proof: Let $V^*(x)$ be the optimal cost of (9a). We prove the claim by showing that the optimal cost between two time instances is upper bounded by zero, i.e.,

$$V^*(x_{k+\sigma_\mu}) - V^*(x_k) \leq 0. \quad (13)$$

To show this, consider the shifted candidate solutions (11) and (12) from Theorem 1. It suffices to prove this claim for $\sigma_\lambda = 1$, where we consider the cost $V(x^*_{1|k}, \mathbf{u}_{k+1})$ of (9) at time $k + 1$ under $\mathbf{u}_{k+1}$. Then by optimality

$$V^*(x_{k+\sigma_\mu}) - V^*(x_k) \overset{\sigma_{\mu=1}}{\leq} V(x^*_{1|k}, \mathbf{u}_{k+1}) - V^*(x_k).$$

Substituting the individual terms of the cost function yields

$$\begin{aligned}
&V(x^*_{1|k}, \mathbf{u}_{k+1}) - V^*(x_k) \\
&= V_f(x^*_{N|k+1}) + \sum_{i=0}^{N-1} l(x^*_{i|k+1}, u^*_{i|k+1}) \\
&\quad - V_f(x^*_{N|k}) - \sum_{i=0}^{N-1} l(x^*_{i|k}, u^*_{i|k}) \\
&= \|x^*_{N|k}\|^2_{A_K^\top P A_K} + \|x^*_{N|k}\|^2_{Q_K} - \|x^*_{N|k}\|^2_P \\
&\quad - \|x^*_{0|k}\|^2_Q - \|u^*_{0|k}\|^2_R \\
&\overset{(4)}{\leq} -\|x^*_{0|k}\|^2_Q - \|u^*_{0|k}\|^2_R = -\|x_k\|^2_Q - \|u_k\|^2_R.
\end{aligned}$$

Since by Assumption 1 the matrices Q and R are positive definite, we establish asymptotically that

$$\begin{aligned}
0 &\leq \lim_{t\to\infty} \frac{1}{t} V^*(x_t) - V^*(x_0) \\
&\leq \lim_{t\to\infty} \frac{1}{t} \sum_{k=0}^{t} (\|x_k\|^2_Q + \|u_k\|^2_R) = 0
\end{aligned}$$

which concludes the proof. ■

Remark 1: The above considerations are valid for any MPC setup involving the design of a terminal region and controller (not only the linear one). Here, we consider only the linear case to provide concise results and clearly show which elements of the proofs are changed and which are kept unvaried with respect to a standard MPC setup.

TABLE I

PERFORMANCE OF FIXED VS MULTI-MODE CONTROL.

Fig. 3: unlimited res.	state err.	act. energy	res. cost
— fixed mode 3	0.466	0.362	**1.00**
— fixed mode 1	**0.441**	**0.212**	5.00
— multi-mode	**0.442**	**0.214**	**2.19**
Fig. 4: limited res.	**state err.**	**act. energy**	**res. cost**
— fixed mode 3	0.466	0.362	1.00
— fixed mode 1	0.497	0.163	–
— multi-mode	0.446	0.236	1.76

C. Criteria for Mode Switching

The number of modes M is decided based on the target requirements for flexibility, and the assigned budgets depend on the computational power of the target platform, paired with the target periods and deadlines to satisfy. The expected performance of the modes can be characterized offline or estimated online based on the current MPC prediction. Instead, the cost of the budget allocation could be evaluated by analyzing the evolution of free resources. Consistently with our choice of switching modes before a new period starts, it is reasonable to run this analysis over the last period and update the budget consequently.

Overall, the choice of the new mode and budget is likely to be formulated as a separate optimization problem. Its objective function will be based on an integral of (7), to account for resource cost, and (9a), for control performance, with proper weights based on the system properties.

A practical advantage of the proposed approach is that the same optimization problem (9) can be implemented across all modes, by properly adapting σ_μ in (9c). Thanks to the shared time base and the fixed prediction horizon N, in practical implementations when using e.g., CasADi, there is no need to recompile the control function when switching between modes, providing a seamless and lightweight multi-mode formulation. The only difference between the modes is the sequence of control input values at the beginning of the prediction, which is passed as an input parameter at runtime.

IV. SIMULATION ANALYSIS

To motivate the proposed approach, we set up a simulation using a simple cart-pole system as an example application. The state variables $x = [p, \theta, v, \omega]^\top$ represent the linear and angular position, and velocity of the cart and the pole, respectively. The control input u is a force applied horizontally to the cart. The system is linearized about the unstable vertical configuration with matrices

$$A_c = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 327/200 & 0 & -25/24 \\ 0 & 981/40 & 0 & -375/24 \end{bmatrix}, \quad B_c = \begin{bmatrix} 0 \\ 0 \\ 35/18 \\ 25/6 \end{bmatrix}.$$

We choose unit weights $Q = I$, $R = I$ and $W_r = 1$ and use the same $\beta_\mu = 1$ for all modes. Starting from the vertical configuration, with cart position -0.4 m, the controller is required to track a step change in the reference position and stabilize it at a target position $+0.4$ m. Once stabilized, an impulse disturbance force is applied to the cart (at $t \approx 2.8$ s), which the controller is required to reject. We define a safety range for the cart position $p \in [-0.75, +0.75]$ m and the actuator force $u \in [-10, +10]$ N, and we impose them as constraints of (9). We define a discretization step of $h = 20$ ms and define three control modes with period $h_1 = h$, $h_2 = 2h$, $h_3 = 5h$ and a prediction horizon of $N = 80$ steps. The results of the experiments are reported in Table I.

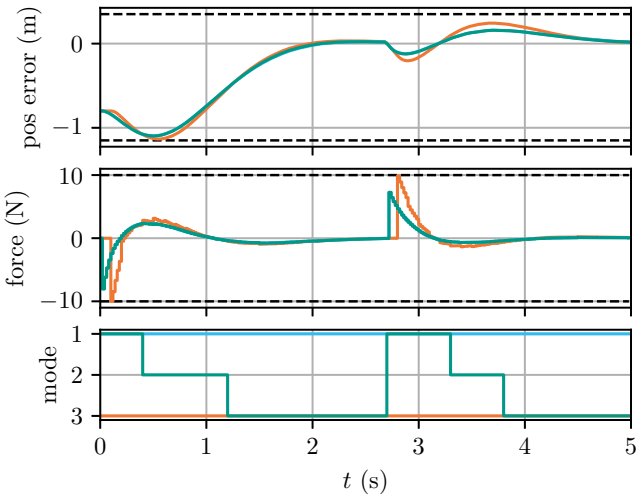

Fig. 3. Testing classic fixed-mode designs using the slowest or fastest mode (orange and blue, respectively), against the proposed multi-mode approach (green). In the first two plots, the blue and green trajectories nearly coincide.

Fig. 3 shows the tests comparing our multi-mode approach against classic single-mode designs. We consider three setups: A fixed-mode design with slowest mode (mode 3), a fixed-mode design with fastest mode (mode 1) and a multi-mode design with all 3 modes. In each setup, the performance of the closed-loop system is measured as mean squared error of the normalized state and input trajectory, as well as the minimum total amount of resources required to run each setup over the simulation.

The slowest mode (orange line) fails to provide a satisfactory behavior: the controller response is delayed, the cart-pole oscillations are larger and the peak of the actuator force is higher. With the fastest mode (blue line), the best possible performance is achieved. To fit the execution of the mode within the shorter period, however, a higher budget is needed, which comes at the expense of higher resource usage costs. The superior performance offered by the fastest mode, on the other hand, is unlikely to be required once the plant conditions are no longer critical. Indeed, approaching steady-state conditions, the difference between the modes is barely noticeable.

In the last setup (green line), the multi-mode approach is used to gradually switch to mode 2 and 3 (which can run on a lower resource level), while the system approaches the steady state conditions. In this example, the multi-mode design achieves a large reduction of the resource costs (approx. 70%), against a small degradation of the control performances ($< 5\%$).

In Fig. 4, we repeat the tests with the same setups as

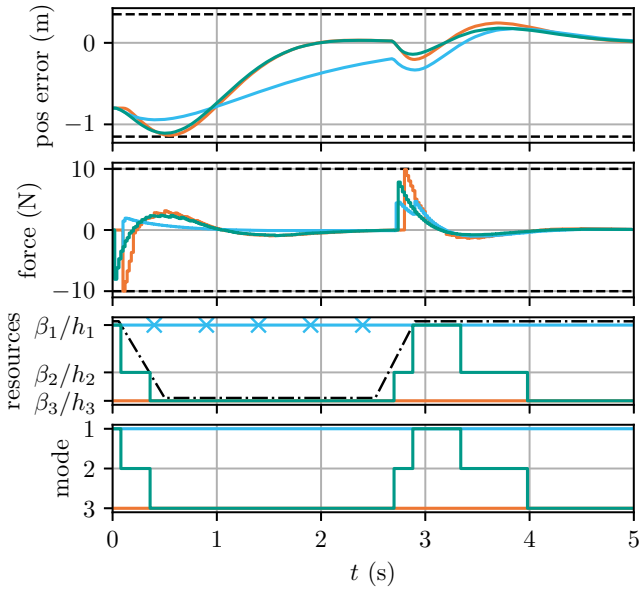

Fig. 4. Multi-mode pattern with limited resource availability (represented by the dash-dotted line). The activation of the faster modes is cut ($t \approx 0.3$ s) or delayed ($t \approx 2.8$ s) under critical conditions if not enough resources are available.

above, but under a limited and variable resource budget. We define an upper bound to the free resources that cannot be exceeded. If such budget is insufficient to execute a mode within the corresponding deadline, a backup LQR controller will be applied in place of the optimal MPC trajectory.

In this setting, the fixed-mode design with the slower mode 3 (orange line) performs as in the previous experiment. With mode 1 (blue line), the performance are worse in this scenario, due to the inability of the control task to execute as expected. Under the reduced resource availability, the backup controller takes over; instability is avoided, but the system response is slower. The switching pattern of the multi-mode design (green line) is readjusted, within this constraint, to adapt to the new resource availability. The resulting trajectory is slightly worse than the one from the previous experiment, but still manages to improve the performance over both fixed setups. Also in this scenario, the multi-mode design manages to cope with the variability of the resources while maintaining good levels of performances.

V. Conclusion and Future Work

The multi-mode MPC approach presented in this paper is a scalable control strategy, where the accuracy and the computational complexity of the controller can be dynamically adjusted at runtime. This feature is essential for a successful execution under variable resource conditions. We showed that our design guarantees feasibility and asymptotic stability of the controlled system under arbitrary switching of modes. Finally, with a simulation analysis we showed the potential of the proposed approach for resource costs reduction, with limited impact on the control performance.

The control and resource formulation we adopted is fairly general and allows us to model resource availability and constraints for different architectures and use cases. Further steps towards a complete resource-aware control framework will cover finding a mechanism for the optimal selection of the modes and allocation of the resources, and extending the allocation problem to the case of multiple control applications concurring for shared resources on the same platform.

References

[1] P. Park, S. Coleri Ergen, C. Fischione, C. Lu, and K. H. Johansson, "Wireless Network Design for Control Systems: A Survey," *IEEE Commun. Surv. Tutor.*, vol. 20, no. 2, pp. 978–1013, 2018.

[2] A. W. Colombo, T. Bangemann, S. Karnouskos, J. Delsing, P. Stluka, R. Harrison, and J. L. Jammes, Francois Lastra, Eds., *Industrial Cloud-Based Cyber-Physical Systems*. Springer Int. Publishing, 2014.

[3] S. M. Salman, V. Struhár, Z. Bakhshi, V.-L. Dao, N. Desai, A. V. Papadopoulos, T. Nolte, V. Karagiannis, S. Schulte, A. Venito, and G. Fohler, "Enabling Fog-based Industrial Robotics Systems," in *Proc. IEEE Int. Conf. Emerg. Technol. Fact. Autom. (ETFA)*, 2020, pp. 61–68.

[4] S. M. Salman, V. Struhár, A. V. Papadopoulos, M. Behnam, and T. Nolte, "Fogification of Industrial Robotic Systems: Research Challenges," in *W. Fog Computing and the IoT (Fog-IoT)*, 2019, pp. 41–45.

[5] W. P. M. H. Heemels, M. C. F. Donkers, and A. R. Teel, "Periodic Event-Triggered Control for Linear Systems," *IEEE Trans. Automat. Control*, vol. 58, no. 4, pp. 847–861, 2013.

[6] D. Antunes and W. P. M. H. Heemels, "Rollout Event-Triggered Control: Beyond Periodic Control Performance," *IEEE Trans. Automat. Control*, vol. 59, no. 12, pp. 3296–3311, 2014.

[7] T. Gommans, T. Theunisse, D. Antunes, and W. Heemels, "Resource-Aware MPC for Constrained Linear Systems: Two Rollout Approaches," *J. Process Control*, vol. 51, pp. 68–83, 2017.

[8] S. Wildhagen, F. Dürr, and F. Allgöwer, "Rollout Event-Triggered Control: Reconciling Event- and Time-Triggered Control," *at - Automatisierungstechnik*, vol. 70, no. 4, pp. 331–342, 2022.

[9] K. Hashimoto, S. Adachi, and D. V. Dimarogonas, "Self-Triggered Model Predictive Control for Continuous-Time Systems: A Multiple Discretizations Approach," in *Proc. IEEE Conf. Decis. Control (CDC)*, 2016, pp. 3078–3083.

[10] Y. Lian, S. Wildhagen, Y. Jiang, B. Houska, F. Allgöwer, and C. N. Jones, "Resource-Aware Asynchronous Multi-Agent Coordination via Self-Triggered MPC," in *Proc. IEEE Conf. Decis. Control (CDC)*, 2020, pp. 685–690.

[11] S. Wildhagen, C. N. Jones, and F. Allgöwer, "A Resource-Aware Approach to Self-Triggered Model Predictive Control," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 2733–2738, 2020.

[12] L. Grüne, J. Pannek, and K. Worthmann, "Ensuring Stability in Networked Systems with Nonlinear MPC for Continuous Time Systems," in *Proc. IEEE Conf. Decis. Contr. (CDC)*, 2012, pp. 14–19.

[13] P.-B. Wang, X.-M. Ren, and D.-D. Zheng, "Robust Nonlinear MPC With Variable Prediction Horizon: An Adaptive Event-Triggered Approach," *IEEE Trans. Automat. Control*, vol. 68, no. 6, pp. 3806–3813, 2023.

[14] P. Patrinos and A. Bemporad, "An Accelerated Dual Gradient-Projection Algorithm for Embedded Linear Model Predictive Control," *IEEE Trans. Automat. Control*, vol. 59, no. 1, pp. 18–33, 2014.

[15] G. Cimini and A. Bemporad, "Exact Complexity Certification of Active-Set Methods for Quadratic Programming," *IEEE Trans. Automat. Control*, vol. 62, no. 12, pp. 6094–6109, 2017.

[16] P. Varutti and R. Findeisen, "Compensating Network Delays and Information Loss by Predictive Control Methods," in *Proc. European Contr, Conf. (ECC)*, 2009, pp. 1722–1727.

[17] K. Zhang, J. Sprinkle, and R. G. Sanfelice, "Computationally Aware Switching Criteria for Hybrid Model Predictive Control of Cyber-Physical Systems," *IEEE Trans. on Automation Science and Engineering*, vol. 13, no. 2, pp. 479–490, 2016.

[18] K. Koichi, "Self-Triggered Model Predictive Control for Linear Systems Based on Transmission of Control Input Sequences," *Journal of Applied Mathematics*, vol. 2016, no. none, pp. 1–7, 2016.

[19] Y. Liu, P. Zeng, J. Cui, C. Xia, and Y. Sun, "A Self-Triggered Approach for Co-Design of MPC and Computing Resource Allocation," *IEEE Internet Things J.*, vol. 11, no. 14, pp. 25 024–25 032, 2024.

[20] P. Skarin, J. Eker, and K.-E. Årzén, "A Cloud-Enabled Rate-Switching MPC Architecture," in *Proc. IEEE Conf. Decis. Control (CDC)*, 2020, pp. 3151–3158.